# LLMs for Lawyers: An Illustrated Guide

legaled.ai/llms-for-lawyers-an-illustrated-guide

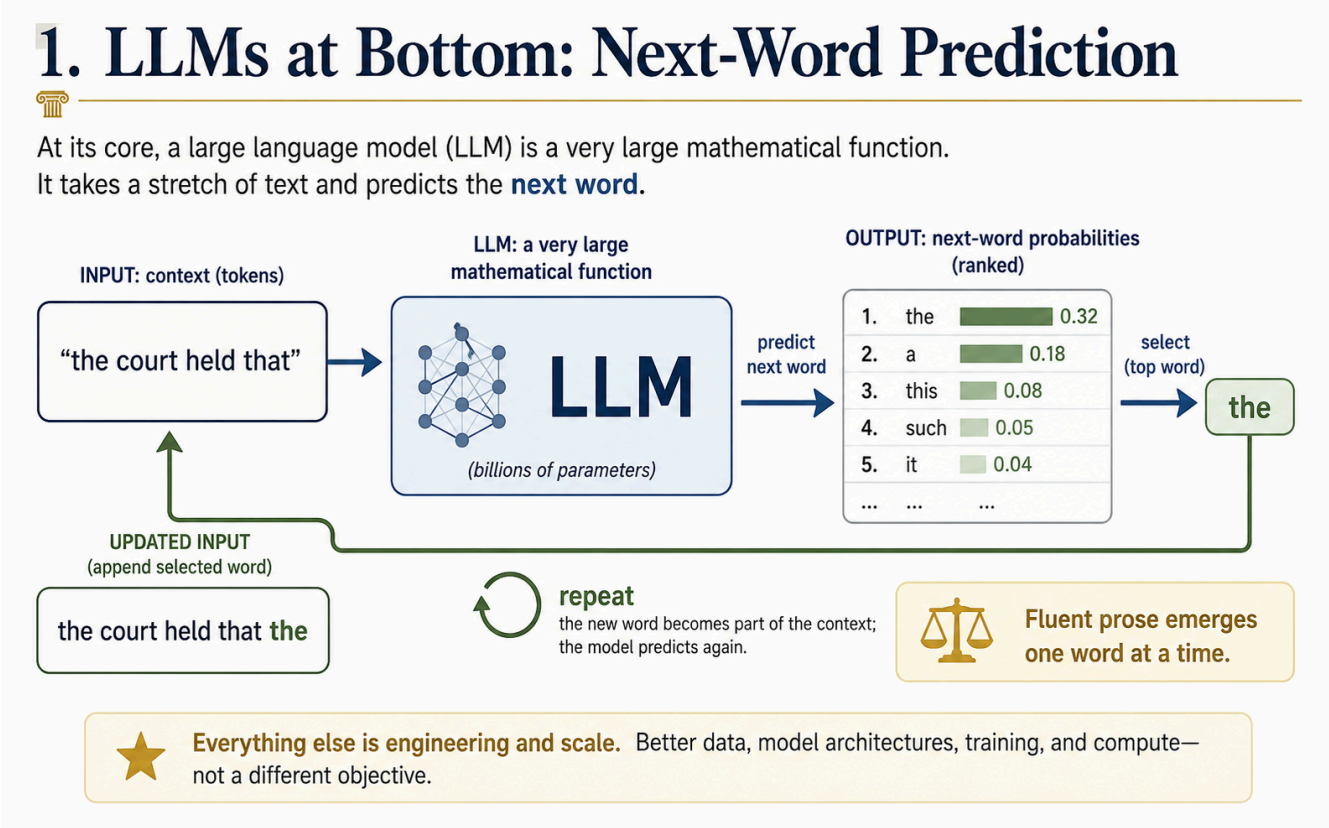
Seth Chandler May 20, 2026

An oversimplified visualization of how large language models work

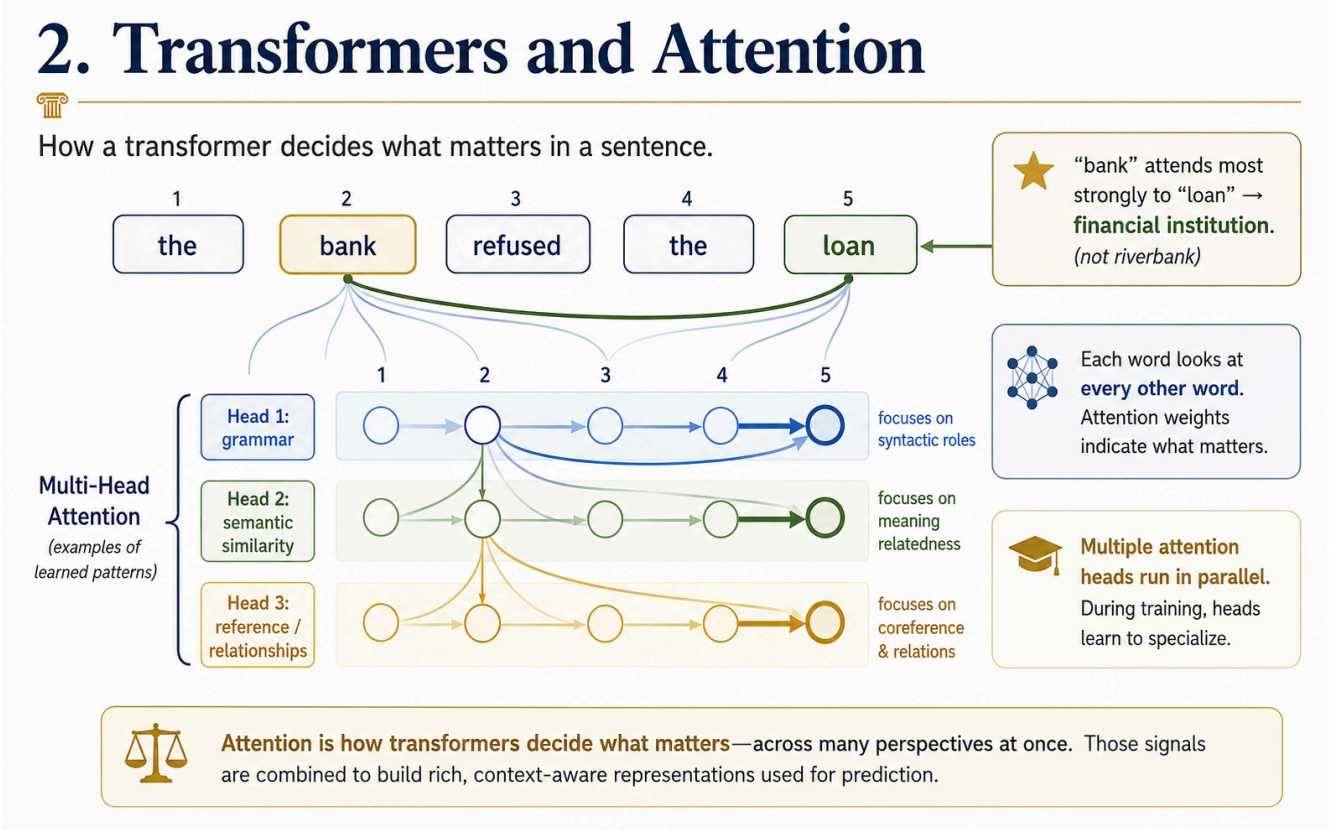
I'm appearing on a panel on AI in a few weeks at the American Society of Law Medicine and Ethics. One of the agenda items is a discussion of how Large Language Models work. I guess there are a lot of people who don't know. And there are even more – including me – who don't understand many of the complications. But I thought I would use Generative AI to build a basic visual guide to how Generative AI works. (I'm treating Generative AI and Large Language Models as synonyms here). Here is the result of about 30 minutes of collaboration with AI. See the "how it was done" appendix if you are interested in the workflow as much as the result. If you want just the visualizations in the form of a PowerPoint, those are also in the Appendix.

## How Modern LLMs Work: A Brief Tour for Law Professors

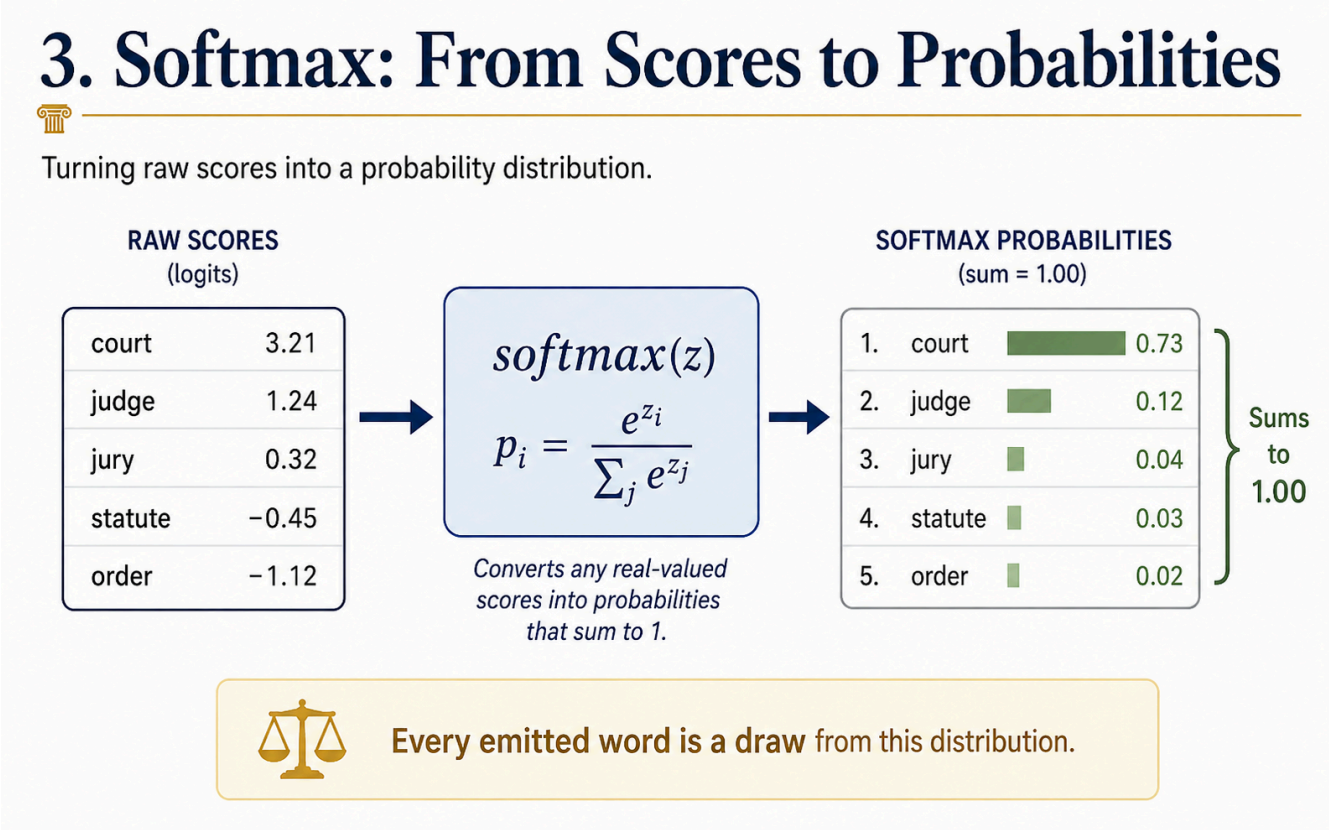
A large language model is, at bottom, a very large mathematical function that takes a stretch of text and predicts what word is likely to come next. Run that prediction over and over, feeding each new word back in, and you get the fluent prose you see in ChatGPT or Claude. Everything else is engineering and scale.



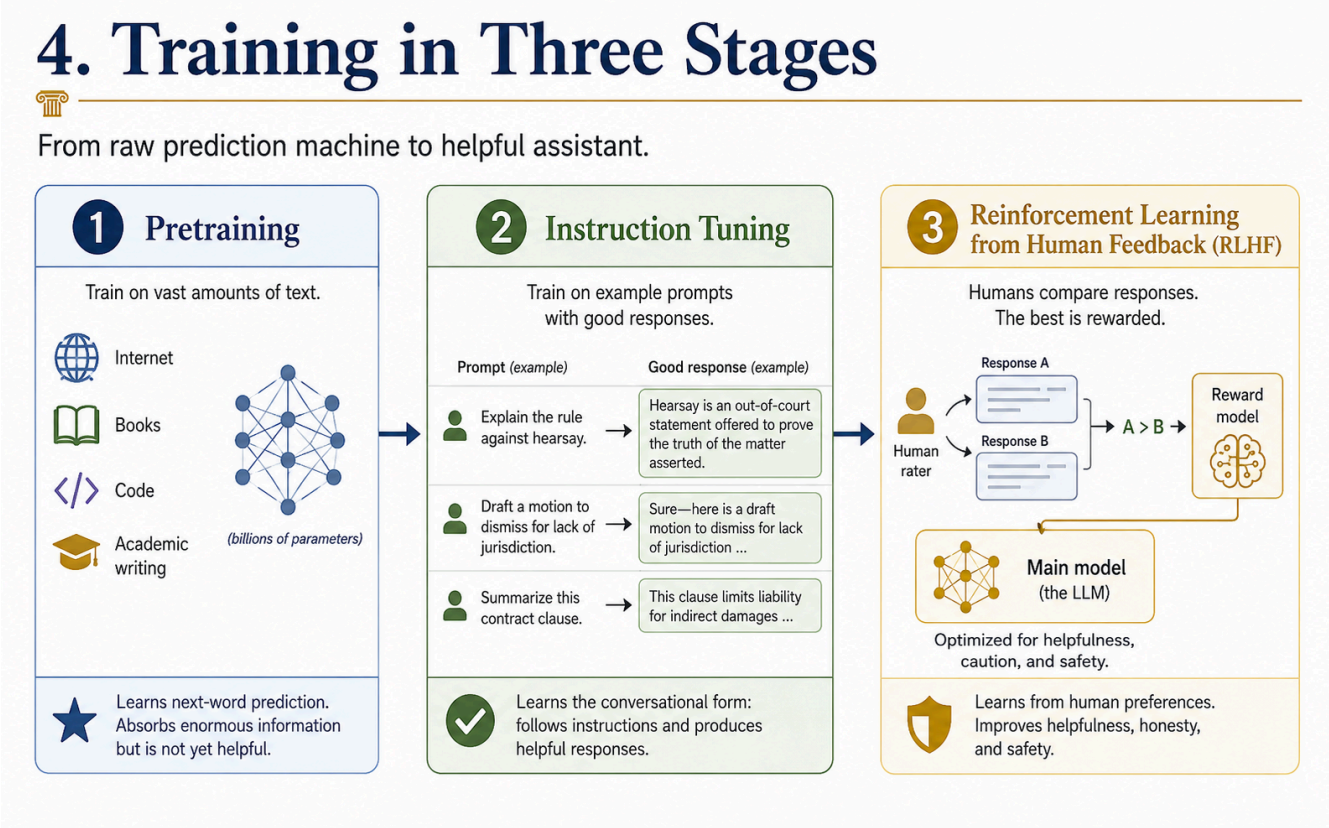

The architecture doing the predicting is called a *transformer*. Its central trick is *attention*, and specifically *multi-headed attention*. When the model processes a sentence, every word looks at every other word and decides how much each one matters for understanding it. In "the bank refused the loan," the word "bank" attends strongly to "loan" and decides it means a financial institution, not a riverbank. "Multi-headed" means the model performs many such comparisons in parallel — one head might track grammatical structure, another semantic similarity, another which pronoun refers to which noun. The model does not know that these are the jobs; the heads simply learn to specialize during training. Attention produces raw numerical scores, which get converted into probabilities by a function called *softmax*.



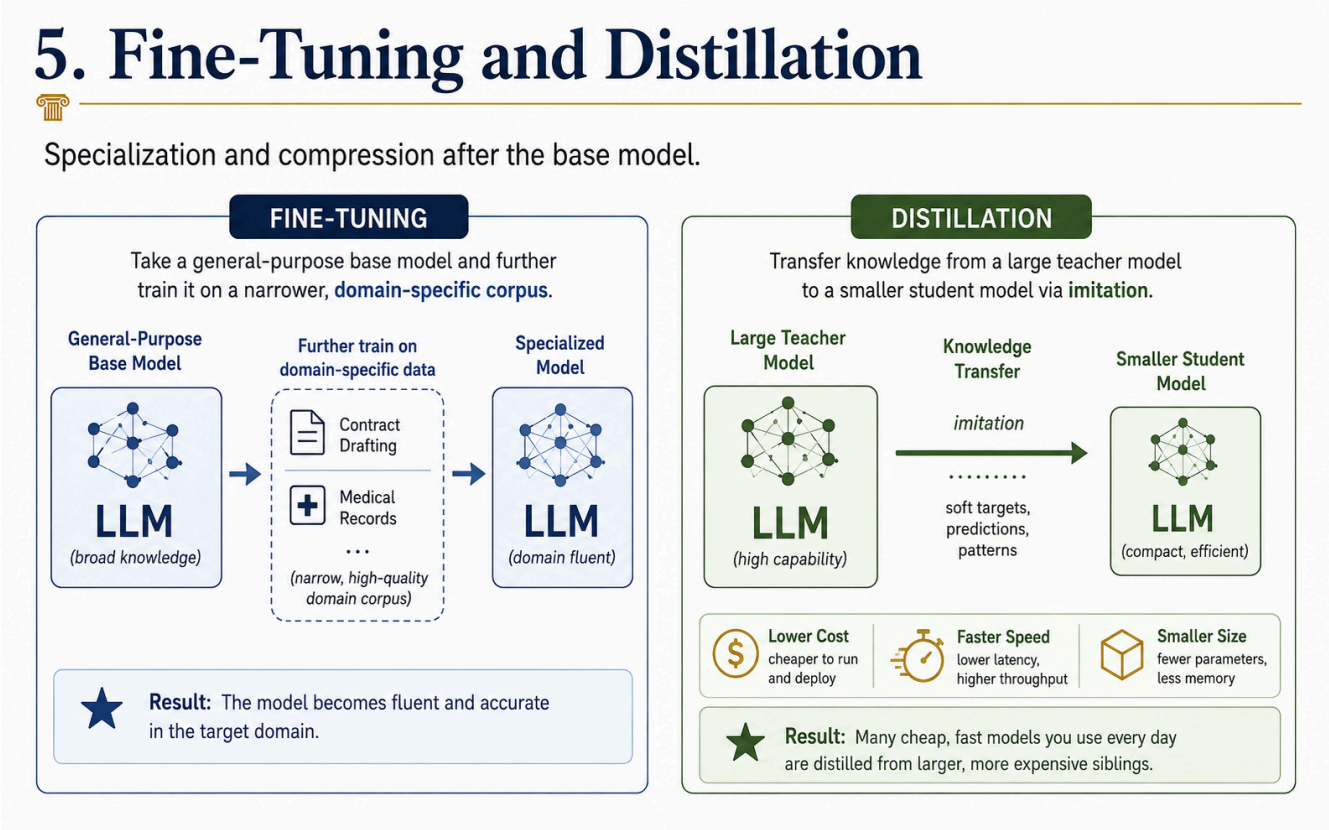

Softmax takes any list of numbers and squashes them into weights that sum to one — a probability distribution. It is how the model expresses "I'm 73% confident the next word is 'court', 12% 'judge', 4% 'jury'…" Every word the model emits is a draw from a softmax distribution.



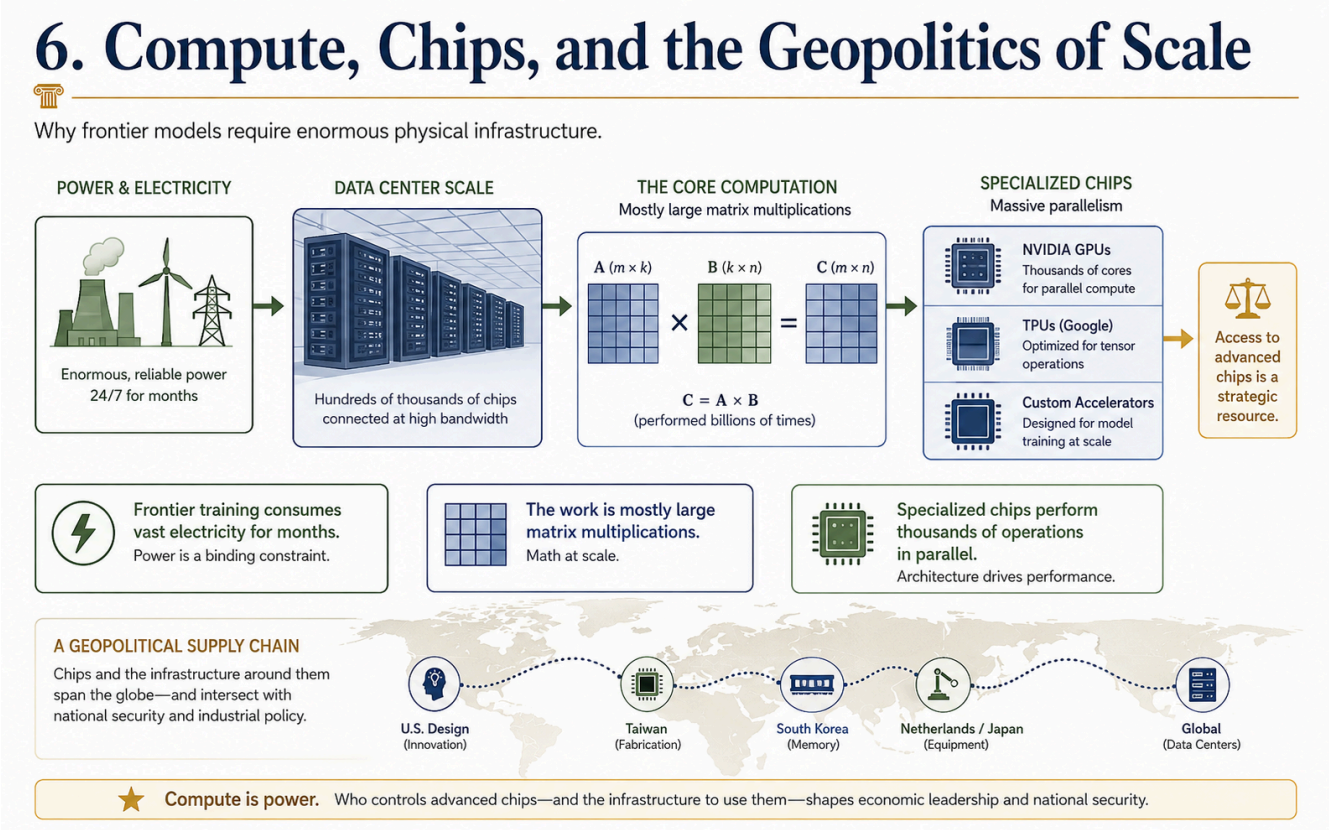

Training proceeds in three stages. *Pretraining* is brute force: the model reads a substantial fraction of the internet, plus books, code, and academic writing, and is rewarded for predicting the next word correctly. This takes months and produces something that has absorbed enormous information but does not yet know how to be helpful — it will happily continue your question with three more questions. *Instruction tuning* fixes this. Human contractors write example prompts paired with good responses, and the model is fine-tuned on these to learn the conversational form. The third stage is *reinforcement learning from human feedback*: humans rate pairs of responses, a separate "reward model" learns to predict those preferences, and the main model is trained to maximize predicted reward. This is what makes Claude polite, ChatGPT cautious, and both reasonably resistant to producing slurs or bomb recipes.



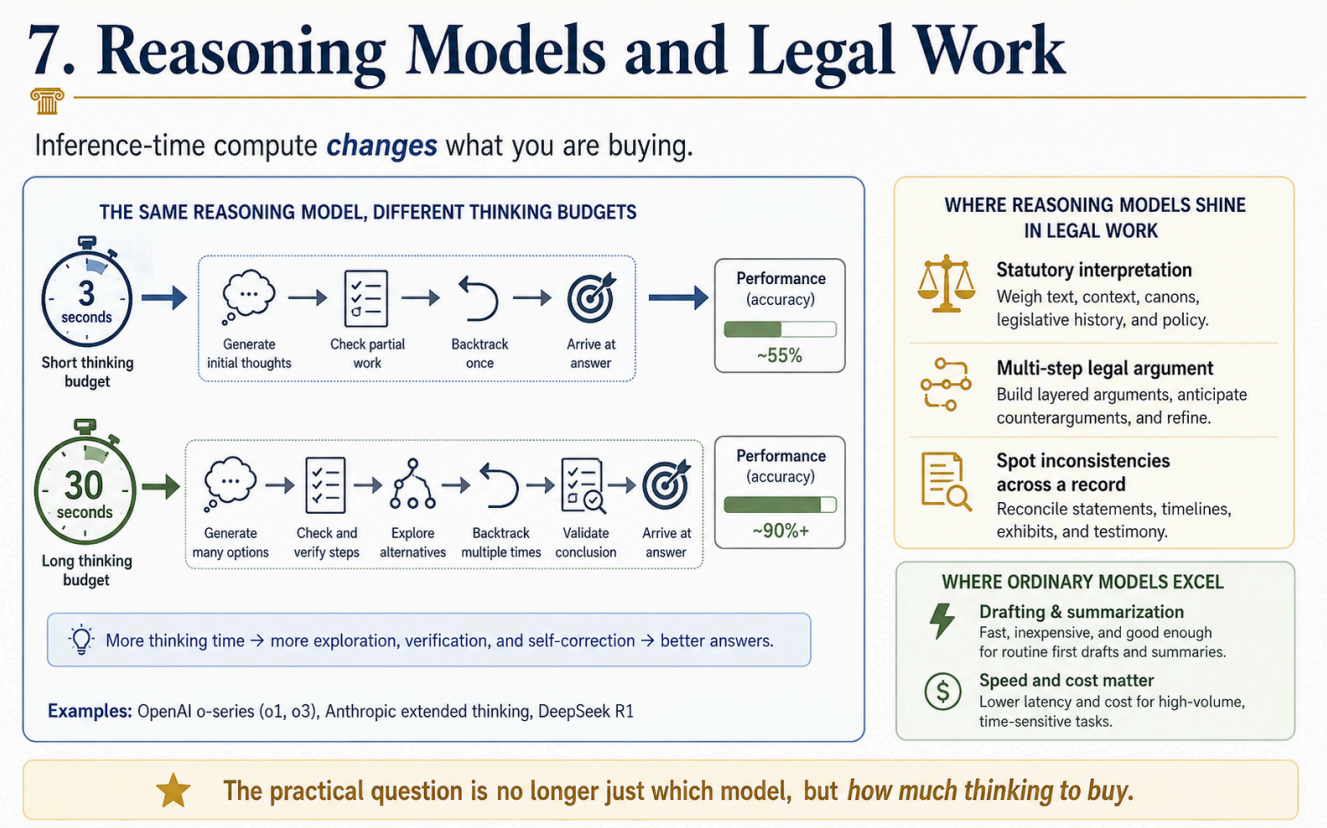

Beyond these basics, models can be specialized through further *fine-tuning* — additional training on a narrower corpus to produce, say, a model fluent in contract drafting or in medical records. *Distillation* is the complementary move: a small "student" model is trained to imitate the outputs of a large "teacher" model, inheriting much of its capability at a fraction of the size and cost. Most cheap, fast models you encounter are distilled from larger siblings.




All of this is staggeringly compute-intensive. Training a frontier model consumes electricity on the scale of a small city for months, which is why the major labs are now building or contracting for dedicated power generation. The computations themselves are overwhelmingly *tensor operations* — large matrix multiplications — and they run on specialized chips, primarily Nvidia's GPUs and increasingly custom accelerators like Google's TPUs, designed to perform thousands of such operations in parallel. Access to these chips has become a strategic and geopolitical resource.




The newest development is *reasoning models* and *inference-time compute*. Until recently, a model's intelligence was fixed at training time. Reasoning models — OpenAI's o-series, Anthropic's extended thinking, DeepSeek's R1 — are trained, often through reinforcement learning on problems with checkable answers like math and code, to produce long internal chains of thought before responding. Performance scales with how long you let them think; compute spent at inference is partially substitutable for compute spent at training. For legal work this matters because reasoning models are markedly better at tasks with structure — statutory interpretation, multi-step argument, spotting inconsistencies across a record — while ordinary models remain faster and cheaper for drafting and summarization. The practical question for any task is no longer just *which model* but *how much thinking* to buy.




## Appendix: How This Post Was Created

This post serves two purposes. First is the obvious one: explain how large language models (AI) works to a non-technical audience. The second is more meta. If you think the post is kind of cool – particularly the visualizations – you might be interested in how to generalize, how to create posts on different topics with a similar visualization-heavy style. This appendix addresses both issues.

First, I just gave Claude (Opus 4.7) a basic question. I probably should have done this in Claude for Word, but habit took me to the regular web interface. It took a little bit of domain knowledge but, as you can see, the prompt was very basic.

> Come up with a very high level description of how modern transformer LLMs work and how they are trained. The audience is non-technical law professors. I do think we need the vocabulary of multi-headed attention and softmax and reinforcement learning and instruction tuning but we can't be going through block diagrams. We also need to discuss the newer development of reasoning models and inference time compute. I am thinking 600 words is about right.

Claude produced something decent but on reading it, I realized it needed a bit more to capture some issues relevant to many law professors.

> Could we add a paragraph on other forms of fine tuning and distillation. Could you also a very brief mention of where the energy costs fit in and the importance of specialized chips to do the computations – mostly tensor calculations – at speed.

Here is Claude's response. I've wrapped it in a bit of HTML code so that it is more attractive when viewed in this blog.

## How Modern LLMs Work: A Brief Tour for Law Professors

A large language model is, at bottom, a very large mathematical function that takes a stretch of text and predicts what word is likely to come next. Run that prediction over and over, feeding each new word back in, and you get the fluent prose you see in ChatGPT or Claude. Everything else is engineering and scale.

The architecture doing the predicting is called a **transformer**. Its central trick is **attention**, and specifically **multi-headed attention**. When the model processes a sentence, every word looks at every other word and decides how much each one matters for understanding it. In "the bank refused the loan," the word "bank" attends strongly to "loan" and decides it means a financial institution, not a riverbank. "Multi-headed" means the model performs many such comparisons in parallel — one head might track grammatical structure, another semantic similarity, another which pronoun refers to which noun. The model does not know these are the jobs; the heads simply learn to specialize during training.

Attention produces raw numerical scores, which get converted into probabilities by a function called **softmax**. Softmax takes any list of numbers and squashes them into weights that sum to one — a probability distribution. It is how the model expresses "I'm 73% confident the next word is *court*, 12% *judge*, 4% *jury*…" Every word the model emits is a draw from a softmax distribution.

Training proceeds in three stages. **Pretraining** is brute force: the model reads a substantial fraction of the internet, plus books, code, and academic writing, and is rewarded for predicting the next word correctly. This takes months and produces something that has absorbed enormous information but does not yet know how to be helpful — it will happily continue your question with three more questions. **Instruction tuning** fixes this. Human contractors write example prompts paired with good responses, and the model is fine-tuned on these to learn the conversational form. The third stage is **reinforcement learning from human feedback**: humans rate pairs of responses, a separate "reward model" learns to predict those preferences, and the main model is trained to maximize predicted reward. This is what makes Claude polite, ChatGPT cautious, and both reasonably resistant to producing slurs or bomb recipes.

Beyond these basics, models can be specialized through further **fine-tuning** — additional training on a narrower corpus to produce, say, a model fluent in contract drafting or in medical records. **Distillation** is the complementary move: a small "student" model is trained to imitate the outputs of a large "teacher" model, inheriting much of its capability at a fraction of the size and cost. Most cheap, fast models you encounter are distilled from larger siblings.

All of this is staggeringly compute-intensive. Training a frontier model consumes electricity on the scale of a small city for months, which is why the major labs are now building or contracting for dedicated power generation. The computations themselves are overwhelmingly **tensor operations** — large matrix multiplications — and they run on specialized chips, primarily Nvidia's GPUs and increasingly custom accelerators like Google's TPUs, designed to perform thousands of such operations in parallel. Access to these chips has become a strategic and geopolitical resource.

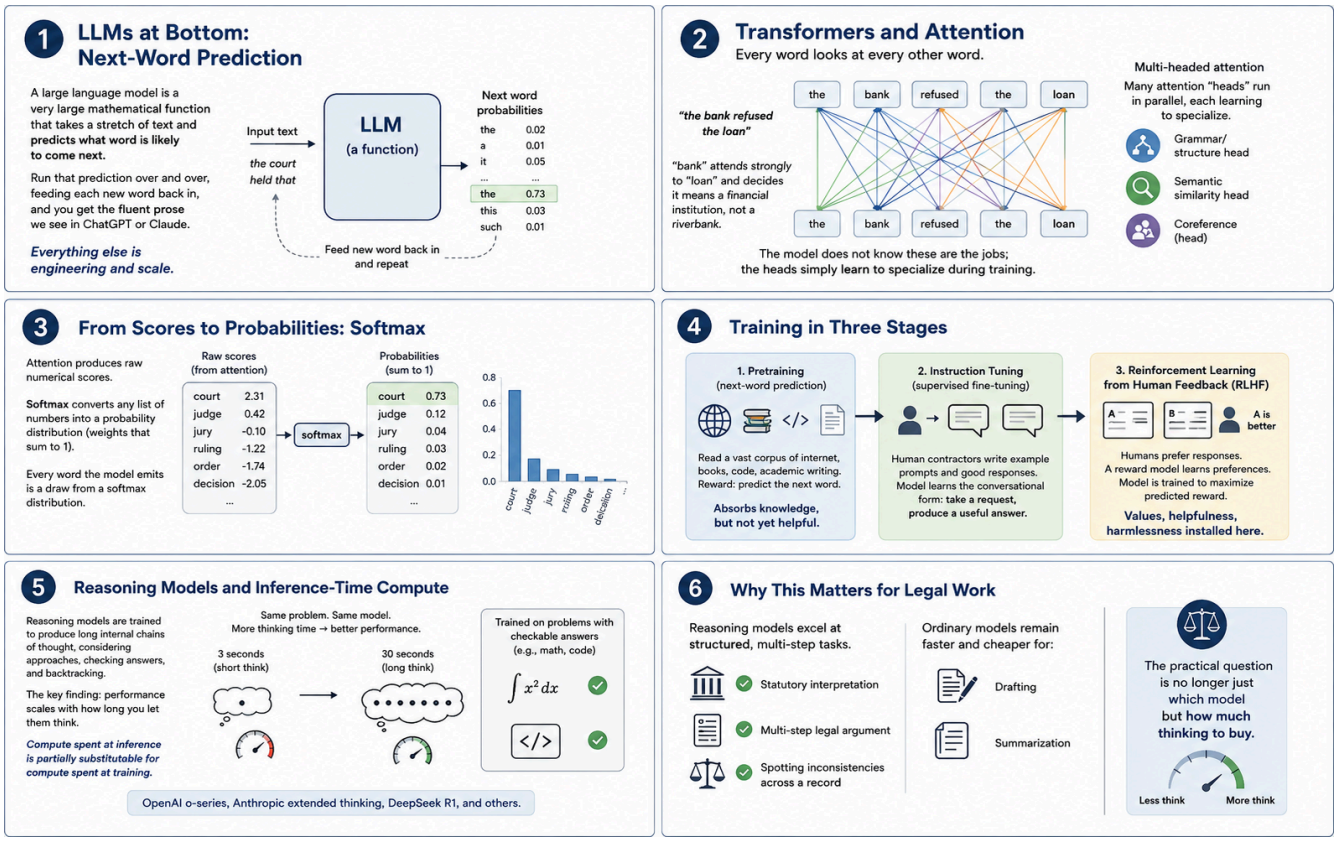
The newest development is **reasoning models** and **inference-time compute**. Until recently, a model's intelligence was fixed at training time. Reasoning models — OpenAI's o-series, Anthropic's extended thinking, DeepSeek's R1 — are trained, often through reinforcement learning on problems with checkable answers like math and code, to produce long internal chains of thought before responding. Performance scales with how long you let them think; compute spent at inference is partially substitutable for compute spent at training.

For legal work this matters because reasoning models are markedly better at tasks with structure — statutory interpretation, multi-step argument, spotting inconsistencies across a record — while ordinary models remain faster and cheaper for drafting and summarization. The practical question for any task is no longer just *which model* but *how much thinking* to buy.

That's a fine lecture but a lot of people are visual learners. Claude doesn't have an image model (yet), so I hopped over to ChatGPT, which now has a fabulous vision model. The prompt was again simple.

> Take this essay and for each paragraph create a graphic explaining what is being discussed. The audience is law professors. Try to capture the ideas involved as much as the details. Each image should have a 16 x 10 aspect ratio. %%% [insert Claude's last response here]

ChatGPT then produced a lovely collage for me.




Unfortunately, I wanted separate images and ChatGPT had responded to the first Claude response, the one that had not included the discussion of electricity use. I thus revised the prompt.

> That's good but we need separate images, not a composite image. Let's try this again. I need 7 16 x 10 images one for each major section of this slightly revised essay.

ChatGPT then responded with the seven images you see shown above. And, basically, that's it. To produce the PowerPoint I just went back to Claude, gave it the seven image files I downloaded from ChatGPT, and told it to make a PowerPoint out of it. (Claude is better than ChatGPT at PowerPoint).

And one final step. Perhaps mistakenly, I originally drafted the blog post inside the ghost editor instead of Microsoft Word. When I was done drafting, however, I just cut and pasted my work into Word (three cheers for Control-a, Control-c), launched the Claude for Word add-in, and used its /copy-edit command to spot typos, grammar errors and the like. It found seven minor errors, none of which my dear readers will now ever see.

And that's pretty much it: about 30 minutes of work, most of which was waiting for ChatGPT to create the seven images. Maybe someone should automate the process with a lecture-to-visualizations skill? Remember, skill files generally work in ChatGPT as well as in Claude.