How Large Language Models Work

What an LLM mechanically is, pretraining and knowledge cutoff, the difference between base and instruction-tuned models, and a first look at in-context learning and context.

Intermediate 20 min

Last updated: August 20, 2026
Read online: https://www.learnforgex.com/en/topics/how-llms-work

---

How Large Language Models Work

"Generative AI" closed with a preview: Large Language Models (LLMs) are generative AI applied to text, built on the transformer architecture, and trained to predict the next token given everything that came before it. That one sentence is accurate, but it compresses a lot -- this lesson unpacks it. By the end you'll know what "pretraining" actually means, why a chat assistant behaves differently from the raw model underneath it, and what "in context learning" is -- a concept that sets up the entire rest of this course, starting with "context" itself in the next lesson.

What Makes a Model a "Large Language Model"?

An LLM is, mechanically, the transformer-based neural network described in "Deep Learning"'s "Common Types of Neural Networks" section, trained on enormous amounts of text with one deceptively simple objective: given a sequence of text, predict the next token. "Large" refers to two things at once -- the size of the network (billions of weights, recall "What Each Node Actually Does" from "Deep Learning") and the size of the training data (a meaningful fraction of publicly available text: books, websites, code, articles). Nothing about the underlying mechanism changed from "Deep Learning" or "Generative AI" -- what changed is scale, and, as the next sections show, scale turned out to produce genuinely new capabilities.

Why One General Model Instead of Many Narrow Ones?

Before LLMs, most NLP (natural language processing) systems were narrow AI in the strictest sense from "Narrow AI and General AI" in the first lesson of this course: a separate model trained for translation, a separate model trained for sentiment analysis, a separate model trained for summarization -- each needing its own labeled dataset and training run. LLMs exist because a single model, pretrained on enough general text with the simple

"predict the next token" objective, turns out to implicitly learn grammar, facts, reasoning patterns, and even how to follow instructions -- well enough that the *same* trained model can translate, summarize, answer questions, and write code, without being separately trained for each task. This is the practical payoff of scale: one general-purpose model replacing what used to require many narrow, task-specific ones.

Pretraining: Learning From the World's Text

Pretraining is the (extremely expensive, usually months-long, run on thousands of GPUs) training phase described generally in "Machine Learning"'s "Training vs. Inference" -- applied here at a massive scale. The model is shown enormous amounts of text and, for each piece, asked to predict the next token; when it's wrong, its weights are adjusted (recall "backpropagation" and "gradient descent" from "Deep Learning") to make that prediction slightly better next time. Repeated over trillions of tokens, this process is what teaches the model grammar, facts about the world (up to whenever its training data was collected -- its **knowledge cutoff**), common reasoning patterns, and even programming languages -- purely as a side effect of getting better and better at predicting what word comes next.

> A model's knowledge cutoff explains a common source of confusion: an LLM can't know about events after its training data was collected, no matter how confidently it answers. This isn't a bug to be patched -- it's a direct consequence of how pretraining works. The "Tools & MCP" category later in this course covers how systems give models access to current information anyway.

Base Models vs. Instruction-Tuned (Chat) Models

A model straight out of pretraining -- often called a **base model** -- is only good at one thing: continuing text in a statistically plausible way. If you give it "The capital of France is," it continues fluently. But if you give it an instruction like "Summarize this email," a base model might just as easily continue with a *list of other instructions* (because that pattern also appears in its training text) instead of actually summarizing anything. This is why almost every LLM you interact with -- a chat assistant, a coding tool -- has gone through an additional training phase after pretraining, broadly called **instruction tuning** (sometimes combined with a technique called RLHF, Reinforcement Learning from Human Feedback), where the model is further trained specifically to follow instructions, hold a conversation, and behave helpfully rather than simply continuing text. The base model's enormous learned knowledge doesn't go away -- instruction tuning reshapes *how* that knowledge is expressed.

In-Context Learning: How a Model Uses What You Give It

Pretraining happens once, offline (recall "Training vs. Inference" again). But an LLM can clearly adapt its behavior *within a single conversation* -- you can teach it a made-up word,

give it an example format to follow, or correct a mistake, all without retraining anything. This is called **in-context learning**: the model's weights don't change at all during a conversation (there is no training happening); instead, everything you've provided -- your instructions, examples, and the conversation so far -- is fed back into the model as input on every single response, and the model's existing (frozen) pretrained abilities are what let it recognize and follow patterns present in that input. This distinction matters enormously: the model isn't "remembering" your conversation the way a database would -- it's re-reading the entire available input fresh, every single time it generates a response.

A First Look at Context

That "everything you've provided" from the previous section has a name: **context**. In the simplest sense, the context is all the text the model actually sees before producing its next token -- your instructions, any background information supplied to it, the conversation history, and whatever it has generated so far in its current response. Context is the *only* channel through which an LLM's behavior can be shaped at inference time (recall from "Training vs. Inference" that no learning happens during inference) -- everything an LLM "knows" for a given response either came from pretraining, or is present somewhere in its context right now. This single idea -- that context is the mechanism, not just a UI feature -- is the foundation the rest of this course builds on: the next lesson looks at context's very real size limits, and the "Tools & MCP" category later covers exactly how systems get the *right* information into a model's context at the right time.

Scale: Parameters, Data, and Compute

Recall from "Deep Learning" that a model's **parameters** are its learned weights -- "billions of parameters" means billions of these adjustable numbers. Modern LLMs range from a few billion to over a trillion parameters, trained on trillions of tokens of text, using thousands of GPUs running for weeks or months (recall "Why Deep Learning Took Off in the 2010s" -- the same data-and-compute story, just at a far larger scale a decade later). Researchers have observed fairly predictable **scaling laws**: performance tends to improve smoothly as model size, data size, and compute all increase together -- which is part of why the industry has pushed toward ever-larger models, though scale alone doesn't solve every limitation (see "LLM Capabilities and Limitations," the last lesson in this category).

Best Practices

- When evaluating a claim about what an LLM "knows," ask whether it came from pretraining (and check the knowledge cutoff) or from context you provided -- these are the only two sources (see "A First Look at Context").
- Remember that a model's weights are frozen after training -- nothing you say in a conversation permanently changes the model, only what's in that conversation's context (see "In-Context Learning: How a Model Uses What You Give It").

- When something behaves unexpectedly instruction-follow-wise, consider whether you're interacting with something closer to a base model's raw completion behavior versus a properly instruction-tuned assistant (see "Base Models vs. Instruction-Tuned (Chat) Models").

Common Mistakes

- **Assuming an LLM "remembers" previous conversations the way a person or a database does.** As "In-Context Learning: How a Model Uses What You Give It" explained, nothing is stored between separate conversations unless a system explicitly re-supplies that information as context in a new one -- the model itself retains nothing.
- **Expecting a base model to follow instructions well.** As "Base Models vs. Instruction-Tuned (Chat) Models" showed, raw next-token prediction and instruction-following are genuinely different trained behaviors -- a model needs the second training phase to reliably do the latter.
- **Treating "bigger model" as an automatic fix for every problem.** Scale (see "Scale: Parameters, Data, and Compute") improves a lot, but as the next lesson in this category will cover, some limitations come from how LLMs work fundamentally, not from insufficient size.

Summary, Cheat Sheet, and Glossary

Summary

- An LLM is a transformer-based model trained at massive scale to predict the next token in text -- the same generative AI mechanism from the previous category, just applied at far greater scale.
- One general pretrained model can perform many tasks (translation, summarization, coding) that used to each require a separate narrow model.
- **Pretraining** teaches a model grammar, facts, and reasoning patterns purely by having it predict the next token over enormous amounts of text.
- **Base models** continue text plausibly; **instruction-tuned (chat) models** go through an additional training phase to reliably follow instructions and hold conversations.
- **In-context learning** means a model's behavior adapts within a conversation without any weights changing -- everything comes from **context**, the text the model actually sees before generating.
- LLM capability tends to improve with scale (parameters, data, compute), though scale doesn't fix every limitation.

Cheat Sheet

- LLM = transformer + massive scale + next-token prediction.

- Pretraining = the one-time, expensive training phase. Knowledge cutoff = how recent its pretraining data is.
- Base model = raw completion. Instruction-tuned model = follows instructions reliably (via instruction tuning / RLHF).
- In-context learning = adapting behavior within a conversation, no weight changes.
- Context = everything the model sees before its next token -- the only channel for shaping its output at inference time.

Glossary

- **Large Language Model (LLM):** a transformer-based generative AI model, trained at massive scale, that predicts and generates text.
- **Pretraining:** the initial, large-scale training phase where a model learns to predict the next token across enormous amounts of text.
- **Knowledge cutoff:** the point in time after which a model has no information, because its pretraining data was collected before then.
- **Base model:** a model that has only gone through pretraining -- continues text plausibly but doesn't reliably follow instructions.
- **Instruction tuning / RLHF:** additional training phases that shape a base model into one that reliably follows instructions and holds conversations.
- **In-context learning:** a model adapting its behavior within a single conversation based on the provided context, without any weight changes.
- **Context:** all the text a model actually sees before generating its next token -- instructions, background information, and conversation history combined.
- **Parameters:** a model's learned, adjustable weights -- "billions of parameters" describes a model's size.
- **Scaling laws:** the observed pattern that LLM performance tends to improve predictably as model size, data, and compute increase together.