Engineer

Large language model

a neural network trained on text to predict the next token

What it is

A large language model is a neural network trained on a very large body of text to do one thing: given the text so far, give a probability to every token that could come next. A token is a short piece of text, often a whole word, sometimes part of a word or a single punctuation mark. Writing an answer is that one step in a loop. The software picks a token according to those probabilities, appends it to the text, and asks again, until the model produces a token that means stop or a length limit is reached. NIST's profile for generative AI, published in July 2024, puts it in one line: "LLMs predict the next token or word in a sentence or phrase." The abbreviation LLM, which the profile itself uses, stands for the same thing.

The architecture behind the models of recent years is the Transformer, described in a 2017 paper on machine translation. It dropped the recurrent and convolutional layers that earlier models were built from and relied on attention alone, a mechanism that lets every position in the text look directly at every other. That made training far easier to spread across many processors, and the paper's model trained in much less time than its predecessors.

"Large" refers to the number of parameters, the weights the network learns during training. A 2020 paper described a model with 175 billion of them, ten times more than any earlier language model that was not sparse, and showed that at that size a model could take on a new task from a handful of worked examples placed in its input, with its weights left exactly as they were.

A model trained only to continue text does exactly that: it continues the text, which is not the same as doing what it was asked. A 2022 paper described the step that turns such a model into an assistant. People wrote the answers they wanted to a set of requests and the model was fine-tuned on them; then people ranked the model's own answers, and it was trained further on those rankings, a

method called reinforcement learning from human feedback. In the paper's evaluations, people preferred the answers of a 1.3-billion-parameter model trained this way to those of the original model with a hundred times as many parameters.

Use the concept in a decision

Put "What is a large language model?" to work in a real decision. We can help relate the concept to the system you are building or buying. An agent wired to the systems you use, tested on cases your own experts write, overseen by your people and never tied to one model.

Explore our services Get in touch →

Test the assumptions behind the term

Use the explanation and linked work to frame questions for your project. We can examine the assumptions with you before a tool is chosen. Built a 268-check guard engine for the repo — 268 commit checks, and 146 tests of the checks themselves

Built a 268-check guard engine for the repo Get in touch →

Why it matters

A language model takes on work that is difficult to cover with fixed rules: sorting a free-text email into the right folder, pulling the fields out of a scanned delivery note, drafting a reply in the customer's own language, summarising a long thread. Work that used to need a person reading every item can now be drafted by software and checked by a person.

The property that makes it useful is also what makes it hard to build on. Its output is usually sampled, so the same input can give different answers on different runs, and it is fluent whether or not it is right. NIST calls confidently stated wrong output confabulation, better known as hallucination, and describes it as a natural result of how these models are built: they produce text that matches the statistics of the text they were trained on, and that is often accurate and sometimes not. NIST also notes that a model can confabulate the reasoning or the citations that seem to justify an answer. A wrong answer looks exactly like a right one, so the only way to know how often a model is right on your work is to measure it on your work.

Nor is there one model to commit to. New models appear regularly, two of them can behave quite differently on the same input, and the provider of the one an application calls decides when it changes. A team building on a language model is building on a part it does not control, and the work that lasts is everything around the model: the cases that measure it, the tools it may call and the rules for what it may do on its own.

NIST's profile names twelve areas of risk for generative AI, among them confabulation, data privacy, information security and intellectual property. It was written for organisations managing risk rather than for developers, but it is a usable checklist for a team deciding what a feature built on a model may touch.

What it looks like in practice

A feature built on a language model starts with written cases, not with a prompt. Each case is a real input and the answer a person who knows the work would accept, kept in a file under version control beside the code:

```
# cases/delivery-notes.yaml
- id: note-0142
  input: scans/note-0142.txt
  expect:
    supplier: "Example Freight"
    reference: "DN-58213"
    pallets: 6
- id: note-0157
  input: scans/note-0157.txt
  expect:
    supplier: "Sample Haulage"
    reference: "DN-59002"
    pallets: 14
```

Which model handles which kind of job is a setting, so a small, fast model can do the extraction while a larger one drafts the replies that need judgement:

```
# routing.yaml
jobs:
  extract-delivery-note:
    model: small-fast
    cases: cases/delivery-notes.yaml
  draft-claim-reply:
    model: large
    cases: cases/claim-replies.yaml
```

Every change runs against all the cases before it ships, whether the change is a new prompt, a new tool or a new model, and the result is a score for each case, kept over time so a fix for one case cannot quietly break another. When a new

model is released, trying it is a one-line change to `routing.yaml` and a run against the cases, and the decision to switch rests on the scores rather than on the announcement.

That is how this site describes its custom AI agents. The choice of model is configuration, not code, with a small, fast model for sorting and extracting and a large one for judgement. An engagement begins by writing down thirty to fifty real examples with the right answers, and every change, whether to a prompt, a tool or the model, runs against all of them before it ships. The page also says that the model is kept replaceable, and that the code, the cases and the data belong to the client, so the agent stays theirs when the model under it changes.

Where it goes wrong

Confident answers that are wrong. A model asked about a contract clause that is not in front of it may describe one anyway, in the same assured tone it uses for a correct answer, and may cite a source that does not exist. Give the model the documents it should answer from, ask it to cite them, and check in code that every citation points at a real document and passage. Measure the error rate on written cases, and route answers with consequences to a person before anything acts on them.

Instructions hidden in the data. An application that lets a model read email, web pages or uploaded files lets the authors of that text speak to the model. A 2023 paper showed this against real applications: instructions planted in content the model was likely to read made it leak data and change what the application did, and NIST's profile lists indirect prompt injection among the security risks of generative AI. Treat everything the model reads as untrusted input, enforce permissions in the tools rather than in the prompt, give the tools only the rights of the person the model acts for, and require a person's approval for any action that cannot be undone.

A model that changes under you. A new version, or a different provider, can answer the same input differently, and an answer that passed last month fails today with no change to your own code. Name an exact model version wherever the provider offers one, and run the whole case file on every change, a change of model included, before it reaches users.

Output that code cannot parse. A model asked for JSON returns JSON followed by a friendly sentence, leaves out a field, or writes a number as a string. Describe the expected shape in JSON Schema, check every answer against it, and treat an answer that fails as an error, to be retried or handed to a person, never as data.

Personal data in prompts and logs. Whatever is sent to a hosted model leaves your own systems, and a log of prompts and answers keeps personal data long after the request is done. NIST adds that models can repeat sensitive material from their training data, a problem it calls data memorisation. Send only the fields a job needs, keep prompt logs under the same retention and access rules as the data in them, and read the provider's terms on retention and training before any personal data reaches it.

Further reading

- NIST AI 600-1: Generative Artificial Intelligence Profile — NIST's July 2024 companion to its AI Risk Management Framework: twelve risks of generative AI, confabulation among them, and suggested actions for each.
- Attention Is All You Need — the 2017 paper that introduced the Transformer, the architecture behind today's language models.
- Language Models are Few-Shot Learners — the 2020 paper on a 175-billion-parameter model that takes on new tasks from examples in its input.
- Training language models to follow instructions with human feedback — fine-tuning on written answers and ranked outputs, the step from text predictor to assistant.
- Not what you've signed up for: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection — instructions planted in data a model reads, demonstrated against real applications.



Glossary

Choose for your actual constraints

Describe the workflow, the existing interfaces and the users who need the change. We can define a testable delivery and a handover that lets your team maintain it.

See what it costs Get in touch →

Make the terminology practical

Ask how “What is a large language model?” applies to your organisation. A concrete example from your current system is enough to start the discussion.

Explore our services

welcome@engineer.company +45 50 10 61 84 Copenhagen, Denmark

Take this page with you

Contact card

Carefully Crafted With Love by D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – present · Engineer ApS · All rights reserved.

Printed from https://engineer.company/glossary/large-language-model/