START HERE

How a Language Model Actually Works

The machinery under the chat box, explained without matrices.

Series editor: Dr Mustafa Mun

Draft, open for academic review

Addalu

First edition, September 2026

START HERE

How a Language Model Actually Works

The machinery under the chat box, explained without matrices.

Series editor: Dr Mustafa Mun

Addaly

First edition, September 2026 · Draft, open for academic review

How a Language Model Actually Works

First edition, September 2026, build 62a26075.

Written by AI models to a written standard, with Dr Mustafa Mun as series editor. Exam keys checked blind by a second model: 3,665 of 3,666 agreed. Academic review invited.

The manuscript was drafted with the assistance of a large language model and was edited and published under his name. That is stated plainly here rather than left to be discovered, because a reader is entitled to know how a book they are trusting was made.

How to cite: *Mun, M. (2026). How a Language Model Actually Works. Addaly. addaly.com/learn/how-llms-work.*

This book is the printed form of the course of the same name at addaly.com/learn/how-llms-work. The website is the living version and is corrected first. Where the two differ, the website is right.

Free to read, free to download, free to print, and free to teach from. There is no paid edition and there never will be.

Every diagram in it is drawn from data by code, not generated as a picture, so the numbers on the page are the numbers in the argument. The answers to every check, test and examination question are at the back.

8 modules · 78 lessons · 29 figures · 8 case studies · 61,047 words · about 11 hours

Brief contents

1

MODULE 1

From characters to vectors

Trace a piece of text from characters to token ids to vectors and back to a score for every possible next token, predict where a tokenizer will split unfamiliar input, and count the parameters of a model from four numbers on its config page.

Lesson 1 · 8 min

What a language model actually is

THE ONE THING TO KEEP

A language model is a function from a sequence of tokens to a probability distribution over the next token; the chat interface, the persona and the tools are all layers built on top of that one function.

One function, called repeatedly

Strip away the chat window, the name, the personality and the tools, and a language model is a single mathematical function. It takes a sequence of integers. It returns a list of numbers, one per entry in its vocabulary, saying how likely each entry is to come next. That is the whole thing.

Written out, the shape is this:

```
logits = model(token_ids)   # in: 1,842 integers.  out: 128,256
numbers
```

Nothing else goes in. There is no separate database being consulted, no lookup of who you are, no second system deciding whether the answer is true. Every capability the model appears to have — translation, summarising, writing code, refusing a request — has to be expressed through that one output.

Most people's mental model is a search engine with a friendlier voice. The distance between that picture and the function above is the reason so many behaviours look mysterious. Once you hold the function in mind, a surprising number of "why did it do that" questions answer themselves.

Deterministic in, distribution out

The function itself is deterministic. Feed identical integers to identical weights and you get identical logits, every time. The randomness you experience in a chat comes from what happens next: something has to pick one token from that distribution, and the picking is usually random by design. Lesson by lesson, keep those two apart. The model produces a distribution. The *sampler* produces a word.

This also means the model does not "choose to answer". It produces a shape over 128,256 possibilities in which some are heavily favoured. A refusal is a distribution where "I" and "can't" are near the top. A hallucinated citation is a distribution where a plausible-looking author name is near the top. Same machinery, no internal switch between modes.

What "large" refers to

Four numbers describe almost any model of this kind, and you can find all four on a public model's config page:

- **Vocabulary size** — how many distinct tokens exist. Commonly 32,000 to 256,000.
- **Model width**, usually called `hidden_size` or `d_model` — how many numbers represent one token position. 2,048 for a small model, 4,096 for a 7-billion-parameter one, 8,192 and up for the largest.
- **Depth** — how many identical blocks are stacked. 16 to 32 for small models, 80 or more for the largest.
- **Context length** — how many tokens fit in one call.

"Large" in *large language model* refers to the parameter count that falls out of the middle two: the total number of adjustable numbers inside. Seven billion, seventy billion, and upward. Every one of them was set by training, none by hand.

The chat is a costume

You send "Hello". What actually reaches the function is closer to this:

```
<|start|>system
You are a helpful assistant. Today is 6 September 2026.
<|end|><|start|>user
Hello<|end|><|start|>assistant
```

The roles, the system prompt, the boundary markers — all of it is text, converted into the same integers as your message. There is no privileged channel. The model has not been told, in some structural sense, that the system prompt outranks you; it has been *trained* on data where text after those markers is usually followed by compliant text. That distinction will explain several security properties later in the course.

Base models make it obvious

A model that has only been pretrained, before anyone taught it to behave like an assistant, does not answer questions. Type "What is the capital of Peru?" into a base model and a plausible continuation is another question, because in the wild that string appears most often in quiz lists:

```
What is the capital of Peru?
What is the capital of Chile?
What is the capital of Bolivia?
```

It is not being unhelpful. It is doing exactly what it does — continuing text — and the assistant behaviour you are used to was added afterwards, by a training stage covered in module five. You can verify this yourself in a few minutes with any base model on Hugging Face, and it is the single most clarifying thing a newcomer can do.

What to carry forward

The rest of this course is an unpacking of one sentence: *tokens in, a distribution over the next token out, repeat*. Tokenisation is how the integers are made. Embeddings and attention are how the function computes.

Sampling is how a distribution becomes a word. Training is how the weights got their values. Every failure mode in module six is a consequence of that sentence rather than a bug on top of it.

BEFORE YOU MOVE ON

A model is given the same prompt twice in the same session and produces two different answers. Which explanation matches the mechanism?

A. The model has some internal state left over from the first call, which changes the second.

B. The weights are updated slightly by each interaction, so the model is never quite the same twice.

C. The function returned a distribution both times, and a sampler drew a different token from it.

D. The two prompts were tokenised differently because of invisible whitespace.

The answer and why: p. 425

Lesson 2 · 7 min

Tokens, and why a token is not a word

THE ONE THING TO KEEP

A token is a chunk of characters chosen by a compression algorithm, not a word and not a letter.

Your text becomes numbers before anything else happens

When you type "Where is the nearest chemist?" the model does not receive letters. A tokenizer cuts the string into pieces drawn from a fixed vocabulary — usually 50,000 to 200,000 entries — and hands over a list of integers.

Everything after that point is arithmetic on those integers. The model has no other access to what you wrote.

That vocabulary is not a dictionary. It is built by an algorithm, usually byte pair encoding, which starts with single bytes and repeatedly merges whichever adjacent pair is most frequent in a training corpus. Sequences that appear constantly end up as one token. Rare ones stay in pieces.

What follows from that

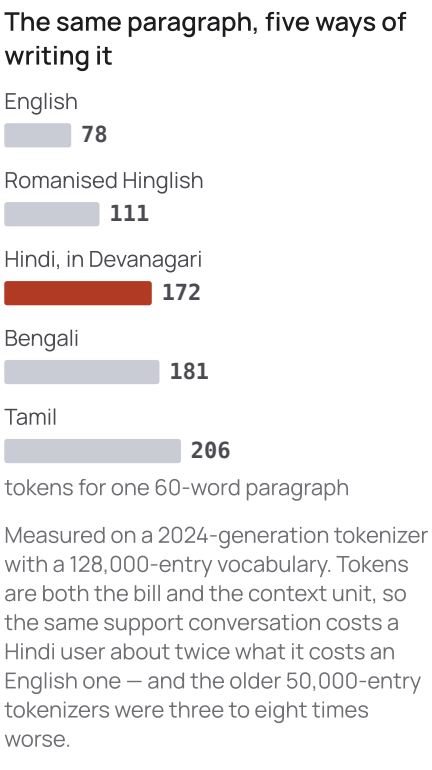
- Common short words are one token each: " the", " and", " is". Note the leading space. It is part of the token.
- A long common word is often one token. An unusual one is several. A surname, a Sanskrit term, or a chemical name may cost six.
- Capitalisation and spacing change the token. "bank", " bank", "Bank" and " Bank" are four different integers.
- Numbers split in ways unrelated to place value. Depending on the tokenizer, 2026 might be one token while 20264 becomes "202" plus "64".

English is cheaper than Hindi, and that is a product decision

Tokenizer vocabularies are built from corpora dominated by English, so English averages roughly four characters per token. Hindi, Tamil, Telugu, Amharic, Thai and Burmese often land near one token per character. The same sentence can cost three to six times more.

Tokens are both the billing unit and the context unit, so this is not a curiosity. A support transcript in Hindi costs several times more to process than the same transcript in English and eats several times more of the context window. If you are choosing a model for a multilingual product, count tokens per language before you compare anything else. Prices quoted per million tokens are not comparable across languages.

Figure 1.1



The strawberry problem

Ask a model how many times "r" appears in "strawberry" and it may say two. This is usually described as a reasoning failure. It is closer to a visibility failure.

The word arrives as something like "str" + "aw" + "berry" — three integers. Nothing in that input exposes individual characters. Whatever the model knows about the spelling of the word, it absorbed indirectly, from text that discussed spelling, hyphenation, rhyme or wordplay. It is second-hand knowledge about a thing it cannot see.

The same explanation covers reversing strings, counting characters, and judging syllables. Meanwhile the model handles far harder semantic work without trouble, which is why the failure feels so strange. If you need character-level work, hand it to code:

```
text = "strawberry"
print(text.count("r"))   # 3, every time
```

Look at your own splits

Every major provider ships a tokenizer you can run locally. Ten minutes with it changes how you write prompts.

```
import tiktoken
enc = tiktoken.get_encoding("cl100k_base")
ids = enc.encode("Where is the nearest chemist?")
print(len(ids), [enc.decode([i]) for i in ids])
```

Run it on a prompt template you use daily. Then run it on the same template translated into a language your users actually speak. The difference is usually larger than people expect.

Why subwords at all

A word-level vocabulary cannot represent anything it has not seen — every new name, typo or product code becomes a single "unknown" token, and the information is gone. A character-level vocabulary handles everything but makes sequences four to five times longer, and the cost of attention grows with the square of length, which you will meet in lesson three.

Subword tokens are the compromise. Any string can be represented, because the fallback is raw bytes, and ordinary text stays short. The price is that the model's view of language is chunked in a way that follows statistics, not meaning or spelling.

BEFORE YOU MOVE ON

A model is asked how many times the letter "r" appears in "strawberry" and answers two. It handles a request to summarise a legal paragraph correctly. What best explains the difference?

A. Counting is arithmetic, and arithmetic is the known weak spot of these models.

B. The tokenizer compresses the word and drops the repeated letters along the way.

C. The word arrives as two or three tokens, so the individual letters are never separately present in the input; whatever the model knows about its spelling is second-hand.

D. There is a gap in the training data — very few documents discuss the spelling of "strawberry".

The answer and why: p. 426

Lesson 3 · 9 min

How the vocabulary gets built, by hand

THE ONE THING TO KEEP

A tokenizer's vocabulary is the frozen output of a compression run over one corpus, so what it splits cleanly is a fact about that corpus and not about language.

Merge the commonest pair, repeat

The previous lesson said the vocabulary is built by byte pair encoding. Here is the algorithm in full, because it takes five minutes and it removes all the mystery.

CASE STUDY · MODULE 1

Two models, one tokenizer, and a Malayalam helpline

An illustrative case, built from documented patterns.

A district collectorate's public grievance cell in Kollam, choosing the model behind an automatic triage and reply-drafting tool

The Kollam collectorate's grievance cell receives roughly 1,900 written complaints a month. About four in five arrive in Malayalam, one in five in English, and a steady trickle in both at once — a Malayalam complaint with an English street address, or an English sentence with two Malayalam words in the middle of it. Three clerks read every one, decide which of eleven departments it belongs to, and draft an acknowledgement. The backlog runs to nine days.

A vendor proposed a triage tool: read the grievance, propose a department, draft the acknowledgement, and let a clerk approve or correct it. Two models were on the table. Model A is an eight-billion-parameter model with a 32,000-entry vocabulary. Model B is nine billion with a 256,000-entry multilingual vocabulary. Their published prices per million tokens were within ten per cent of each other, and the vendor's demonstration — in English — showed Model A ahead by six points on a categorisation benchmark it had prepared.

Anila, the junior engineer in the two-person IT cell, did something nobody had asked her to. She downloaded both tokenizers, took two hundred real grievances out of last year's register, and counted.

The English grievances came out at about 190 tokens each on both models. The Malayalam ones came out at 1,150 tokens on Model A and 470 on Model B. Same complaints, same information, two and a half times the count on the model that had won the demonstration.

That number moved into three other places before she finished the afternoon. The bill: at four in five grievances in Malayalam, Model A would cost roughly two and a half times as much per month for identical work. The context: the vendor's prompt carried twelve past grievances as examples so that the

department assignment stayed consistent between runs, and at Model A's fertility twelve examples plus the new complaint overran the window the vendor had budgeted, so the design would have to drop to three. And the latency: decode produces one token per step, so a 1,150-token draft streams for two and a half times as long as a 470-token one, on a helpline where a clerk is waiting to approve it.

Anila also opened both `config.json` files, because she wanted to know where the extra billion parameters in Model B had gone. Almost all of it was the vocabulary: 256,000 entries at a width of 3,584 is about 918 million parameters in the embedding table, with the same again in the output layer unless the two are tied. Model B was not a bigger model in any sense that mattered to its reasoning. It was the same size of machine with a much larger dictionary bolted on either end — which was precisely the property she cared about.

Mr Pillai, who runs the cell, had to decide, and neither option was free.

Choosing Model B meant overruling the only quantitative measurement anybody in the room possessed. The vendor's six-point lead for Model A was a real number produced by a real test, and the collectorate had nothing to set against it except a token count, which measures cost and not quality. If Model B turned out to be worse at telling a water-supply complaint from a drainage complaint, the clerks would spend their time correcting misfiled grievances and the nine-day backlog would not move.

Choosing Model A meant paying two and a half times over, permanently, in the language four-fifths of the district writes in, and shipping a design with three examples instead of twelve because the window would not hold more.

A third option arrived from the vendor when Anila's numbers reached them: translate every grievance into English with a free Indic translation model, run Model A on the English, translate the acknowledgement back. The token bill collapses. It also adds a second model to the failure path, and the grievances are full of ward names, local place names and community terms that a general translation model handles badly — the exact words on which a misfiled complaint turns.

The meeting ended with the argument unresolved and stated precisely, which was progress: one side had a quality measurement in the wrong language, the other had a cost measurement in the right one.

What would you have done, and what would you have measured first?

YOUR ANSWERS

1. The vendor's benchmark was not dishonest, and it was still the wrong number to decide on. What exactly was wrong with it?

2. Anila's token count affected cost, context and latency at once. Explain the mechanism for each.

3. Model B has a billion more parameters than Model A. Why was that not a reason to expect it to be better at reasoning?

STOP HERE

Write your answers before you turn the page. How it played out starts on p. 50.

CASE STUDY · MODULE 1

How it played out

They spent four days building a Malayalam evaluation set: 150 real grievances pulled from the register, stratified so that all eleven departments appeared, with two clerks independently assigning the department and a third settling disagreements. The clerks disagreed on 19 of the first 50, almost all between two pairs of departments whose boundary had never been written down. Writing that boundary down took a morning and was, by the cell's own account, more useful than the software.

On that set Model B scored 81 per cent on department assignment and Model A scored 74. The vendor's six-point English lead had reversed by seven points in Malayalam. Nobody had lied; the benchmark had simply been in the wrong language, and it took a hundred and fifty labelled items to find out.

They took Model B. The monthly bill came to about a third of what Model A would have cost, because the token ratio and the shorter drafts compounded. The twelve examples fitted.

The translation route was tested anyway, on the same 150 items, and scored 69 per cent — worse than either model directly, and the errors clustered exactly where Anila had guessed: ward names and local terms mangled on the way into English and never recovered.

The habit that survived: the cell now runs the tokenizer over a sample before reading any vendor's price list, and quotes cost per grievance rather than cost per million tokens. Two later proposals were rejected on that number alone.

The questions, discussed

1. The vendor's benchmark was not dishonest, and it was still the wrong number to decide on. What exactly was wrong with it?

It measured the model on English text when four-fifths of the real input is Malayalam. Tokenisation, and therefore how the input is presented to the model at all, differs between the two languages, and so does how much of each language was in the pretraining corpus. A benchmark is a measurement of a

model on a distribution of inputs; change the distribution and the number no longer applies. The collectorate's own 150 labelled Malayalam grievances cost four days and reversed the ranking.

2. Anila's token count affected cost, context and latency at once. Explain the mechanism for each.

Cost, because providers bill per token and the same complaint is more tokens. Context, because the window is measured in tokens too, so a fixed window holds fewer examples when each is longer — which forced the design from twelve examples down to three. Latency, because decode produces one token per step at a roughly fixed rate, so a draft that is two and a half times as many tokens takes two and a half times as long to stream. One measurement, three budgets.

3. Model B has a billion more parameters than Model A. Why was that not a reason to expect it to be better at reasoning?

Because the extra parameters are in the embedding and output matrices, not in the layers that do the work. A 256,000-entry vocabulary at width 3,584 is about 918 million parameters in the lookup table alone, and the same again at the output unless the two are tied. Anila could see this from `config.json` in ten minutes. Parameter count is a shape, not a score, and knowing which part of the shape grew tells you what changed.

MODULE 1 TEST

Check what stuck

6 questions, one right answer each. This one is for you, not for a record. The answers, and the reason behind each, are in the key on p. 424. If two go wrong, re-read the module before the exam.

1 A model insists that "strawberry" contains two letter r. Which account matches the mechanism?

A. The word arrives as about three subword integers, so its letters are not in the input

B. The model has no arithmetic unit, and counting was simply never part of anything it was trained to do

C. Temperature was too high, so a wrong count was drawn from the distribution

D. The tokenizer collapses repeated letters to keep the sequence short

2 Two models are quoted at the same price per million tokens. One has a 32,000-entry vocabulary built mostly on English, the other 256,000 entries covering many scripts. Why are the prices not comparable for a Hindi helpline?

A. Providers apply a surcharge to non-Latin scripts under their published rate cards

B. The larger vocabulary makes each token slower to generate, raising the effective price

C. Hindi requests are silently routed through a translation service that is billed as a separate line

D. The smaller vocabulary produces several times as many tokens for the same Hindi text

3 A config file lists num_attention_heads 32 and num_key_value_heads 8. What has the designer decided?

A. Attention runs in eight sequential passes, so prefill takes eight times as long

B. Thirty-two query heads share eight sets of keys and values

C. The model was trained with eight-way tensor parallelism and cannot afterwards be split any other way

D. Only eight of the thirty-two heads are active for any given token

4 Why can a transformer tell "the river bank" from "the bank refused my loan" when a static word-vector model cannot?

A. The transformer's tokenizer keeps a separate vocabulary entry for every sense a word can carry in a dictionary

B. The transformer looks the word up twice and averages the two results

C. Static models compare with cosine similarity while transformers use Euclidean distance

D. The starting row is the same, and each block adds information gathered from neighbouring positions

5 A base model asked "What is the capital of Peru?" replies with three more quiz questions. What does that demonstrate?

A. The sampler was left at a temperature high enough to produce incoherence

B. The chat template was applied wrongly, so the system prompt never reached the model at all

C. Its knowledge cut-off falls before the question could be answered

D. It is continuing text, and answering is a behaviour added by a later training stage

6 A locally run model produces fluent text, ignores instructions, and writes both sides of the conversation. What do you check first?

A. The embedding table, which may have been truncated when the file was sharded across four parts

B. The chat template and the stop token configuration

C. The temperature, which is most likely above 1.5

D. The quantisation level, because a four-bit file loses instruction-following

Lesson 10 · 8 min

Attention, without a single matrix

THE ONE THING TO KEEP

Attention blends information from earlier positions into the current one; it selects nothing and replaces nothing.

The problem it solves

"The trophy did not fit in the suitcase because it was too small." Which is small? The suitcase. Now change "small" to "large": suddenly "it" is the trophy. A system that processes each word on its own cannot get this. The information that settles "it" sits several words away and flips depending on a word that has not been read yet at that point.

Attention is the mechanism for moving information between positions. Here it is with no matrices.

Questions and adverts

At each position, from the vector currently sitting there, the model computes three smaller vectors:

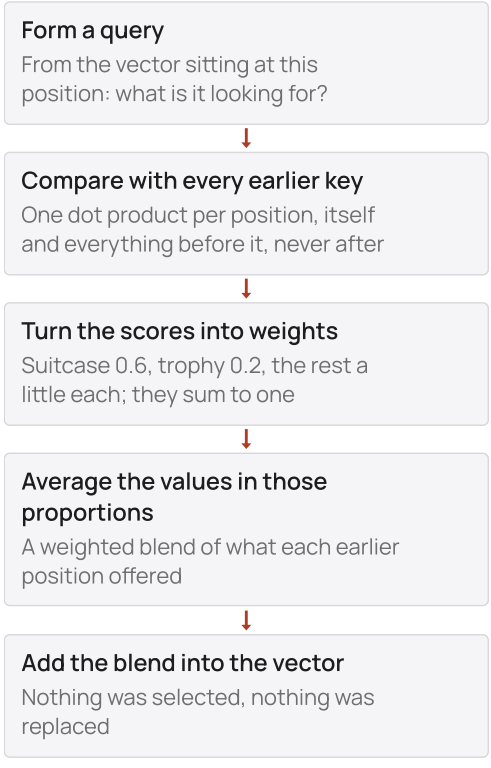
- a **query** — what this position is looking for
- a **key** — what this position offers, an advert for its own contents
- a **value** — the content it will hand over if anyone takes the advert

Now take the position holding "it". Compare its query against the key of every position at or before it. The comparison is a dot product, which is large when two vectors point the same way. Those raw scores are converted into weights that sum to one. In our sentence, "suitcase" might receive 0.6, "trophy" 0.2, and the rest a little each.

Finally, take the weighted average of the value vectors, in exactly those proportions, and add it into the vector at "it".

Figure 2.1

One attention step, at the position holding “it”



Five operations and no decision anywhere in them. The pronoun was not resolved and nothing was recorded; the vector at “it” simply carries a suitcase-flavoured contribution from here on, and every later layer works with the enriched vector.

That is the whole operation. Notice what did not happen. Nothing was selected. The pronoun was not replaced. No decision was recorded. The vector at "it" now carries a suitcase-flavoured contribution, and every later layer works with that enriched vector.

Backwards only

In the models you use for generation, a position may attend to itself and to everything before it, never to what comes after. That restriction is the causal mask.

EXAMINATION

How a Language Model Actually Works — final examination

This paper covers the whole course:

- how text becomes tokens and vectors
- what attention and the feed-forward half each do
- how scores become text
- what the context window is and is not
- where the weights came from
- why the characteristic failures happen
- what a running model costs in memory and money
- what interpretability can and cannot yet establish

56 questions, the whole pool. The site draws 25 for a sitting, pass mark 70%. Work any 25 under your own conditions. Answers from page 478.

1 Module 1 · From characters to vectors

Byte-pair encoding is run on the corpus "low low low lower lowest". Which merge happens first?

A. "l" with "o", because that adjacent pair occurs most often

B. None of them, because the algorithm first has to identify which strings are words before it can merge anything

C. "low", because it is the most frequent whole word in the corpus

D. "est", because it is the most distinctive ending present

2 Module 1 · From characters to vectors

A completion prompt ending "The capital of France is " — with a trailing space — gives worse answers than the same prompt without it. Why?

A. In ordinary text the space belongs to the front of the next token, so you have asked for a rare configuration

B. Trailing whitespace triggers the model's end-of-turn detection

C. The extra token pushes the prompt past a cache boundary, so the prefix has to be recomputed from the beginning

D. The space is stripped by the API, which shifts every position by one

3 Module 1 · From characters to vectors

An embedding search scores "the medicine is safe for children" and "the medicine is not safe for children" at 0.89 similarity. What follows for a retrieval system?

A. Negation should be removed from documents before indexing, so that both forms of a claim map to the same retrievable vector

B. The embedding model is faulty and should be replaced with a larger one

C. Cosine similarity measures what a sentence is about, not what it asserts

D. The threshold should be raised until the two sentences fall on different sides of it

4 Module 1 · From characters to vectors

A model card advertises 128K context and mentions that the model was trained at 8K and extended. What should you expect?

A. The extra context is unusable, since positions beyond 8,192 have no encoding at all

B. Behaviour at 100,000 tokens is a different and usually weaker thing than behaviour at 8,000

C. Prompt caching will not work above 8,192 tokens, because the stored keys and values were computed under the original position scheme

D. Nothing changes, because position interpolation is mathematically lossless

5 Module 1 · From characters to vectors

Somebody proposes fully fine-tuning a 70-billion-parameter model on a laptop. What is the first arithmetic that settles it?

A. The context window would have to be reduced below the length of the training examples

B. 140 GB for the weights alone, and about three times that to train

C. Training compute of 6ND, which no laptop can supply within a working week

D. 70 billion parameters exceeds the largest tensor a consumer GPU driver can address

6 Module 1 · From characters to vectors

A model is described as open. What does that usually mean in practice?

A. The training data, the training code and the weights are all published

B. The weights and a licence are published; the data and pipeline usually are not

C. The model may be used for any purpose, since open is a legal term with a fixed meaning

D. The model file contains the code that runs it, which is what makes it independently auditable

Module 1 · From characters to vectors

The module starts on p. 11.

MODULE 1 TEST

Module 1 test, Q1, p. 52
A model insists that "strawberry" contains two letter r. Which account matches the mechanism?
Answer A: The word arrives as about three subword integers, so its letters are not in the input
Why: The tokenizer cuts the word into pieces like str, aw and berry. Nothing in those three integers exposes individual characters, so whatever the model knows about the spelling was absorbed second-hand from text that discussed spelling. It is a visibility failure rather than a reasoning failure, and the fix is to hand character-level work to code.

Module 1 test, Q2, p. 52
Two models are quoted at the same price per million tokens. One has a 32,000-entry vocabulary built mostly on English, the other 256,000 entries covering…
Answer D: The smaller vocabulary produces several times as many tokens for the same Hindi text
Why: Byte-pair encoding merges whatever was frequent in its corpus, so an English-dominated corpus buys few merges for Devanagari — and a byte-level tokenizer costs three tokens per Devanagari character with no merges at all. The same paragraph is then billed at that ratio, occupies that share of the context window, and streams for that much longer. Count tokens per language before comparing anything else.

Module 1 test, Q3, p. 53
A config file lists num_attention_heads 32 and num_key_value_heads 8. What has the designer decided?
Answer B: Thirty-two query heads share eight sets of keys and values
Why: That is grouped-query attention, and it is a memory decision rather than a quality one. The per-token cache is 2 × layers × kv_heads × head_dim × bytes, so cutting key-value heads from 32 to 8 divides the cache by four — and the cache is the line item that grows with every user, while the weights are shared by all of them.

Module 1 test, Q4, p. 53
Why can a transformer tell "the river bank" from "the bank refused my loan" when a static word-vector model cannot?
Answer D: The starting row is the same, and each block adds information gathered from neighbouring positions
Why: A static model stops at the lookup table, so "bank" gets one vector that is an unhappy average of finance and riverside. A transformer starts from the same row, but attention writes contributions from the surrounding tokens into it layer by layer, so by the upper layers the two senses occupy different regions. An embedding inside a running model is a position that context rewrites, not a label on a word.

Module 1 test, Q5, p. 53

A base model asked "What is the capital of Peru?" replies with three more quiz questions. What does that demonstrate?

Answer D: It is continuing text, and answering is a behaviour added by a later training stage

Why: Pretraining produces a document continuer, and in the wild that string is most often followed by more quiz questions. The habit of answering comes from supervised fine-tuning, and the tone, hedging and refusals come from preference training after that. Neither later stage adds knowledge — all of it, and the cut-off, belong to stage one.

Module 1 test, Q6, p. 54

A locally run model produces fluent text, ignores instructions, and writes both sides of the conversation. What do you check first?

Answer B: The chat template and the stop token configuration

Why: Role markers and end-of-turn markers are ordinary vocabulary entries the model learned from data formatted that way. Use another family's template and it sees a format it never trained on; configure only one of several valid stop tokens and it runs on, helpfully inventing the user's next message. Printing the exact string that reaches the model settles about half of all reports that an open model is bad.

THE LESSON CHECKS

Lesson 1, p. 15

A model is given the same prompt twice in the same session and produces two different answers. Which explanation matches the mechanism?

Answer C: The function returned a distribution both times, and a sampler drew a different token from it.

Why: A language model is a function from a sequence of tokens to a probability distribution over the next token; the chat interface, the persona and the tools are all layers built on top of that one function.

A The model has some internal state…: Conversations feel continuous, so it is natural to imagine the model retaining something between calls; it does not.

B The weights are updated slightly by…: Learning from use is what people assume "AI" means, but the weights are frozen at serving time.

D The two prompts were tokenised differently…: Tokenisation really does depend on exact characters, so this sounds mechanically informed, but identical strings give identical ids.

Lesson 2, p. 19

A model is asked how many times the letter "r" appears in "strawberry" and answers two. It handles a request to summarise a legal paragraph…

Answer C: The word arrives as two or three tokens, so the individual letters are never separately present in the input; whatever the model knows about its spelling is second-hand.

Why: A token is a chunk of characters chosen by a compression algorithm, not a word and not a letter.

A Counting is arithmetic, and arithmetic is…: Counting to three does look like a maths problem, and everyone has heard that models are unreliable at maths.

B The tokenizer compresses the word and…: "Compression" sounds like information is being thrown away, which would neatly explain a missing letter.

D There is a gap in the…: Blaming training coverage explains many real failures, so it is a habit that usually pays off.

Lesson 3, p. 23

Two models both advertise a 128,000-token context window. Your Marathi support transcripts fit comfortably in one but overflow the other. What is the most likely…

Answer D: The two tokenizers were trained on different corpora, so the same Marathi text becomes far more tokens under the one with poorer coverage of the script.

Why: A tokenizer's vocabulary is the frozen output of a compression run over one corpus, so what it splits cleanly is a fact about that corpus and not about language.

A The second model reserves part of…: Some systems really do reserve headroom for the response, but that would not produce a difference of this size between providers.

B The second model's context limit is…: Context is sometimes described loosely in articles, so a unit confusion sounds plausible; in practice both are token limits.

C The transcripts contain formatting that one…: Providers do differ in preprocessing, which makes this feel like a reasonable operational guess.

START HERE

How a Language Model Actually Works

The machinery under the chat box, explained without matrices.

WHAT IT TEACHES	The machinery under the chat box, for people who already use these tools daily and want to know what is actually happening inside them.
WHAT IS INSIDE	Eight modules and 78 lessons, 29 figures drawn from data, eight case studies, eight module tests, a 56-question examination, and every answer with the reason behind it.
LICENCE	CC BY-NC-SA 4.0. Copy, print, share and adapt it for any non-commercial purpose, with credit, under the same licence. There is no paid edition.
THE COURSE	Free to read, with the lessons, the films and the examination, at **addaly.com/learn/how-llms-work**

Addaly

First edition, September 2026 · build 62a26075

Scan to open the course