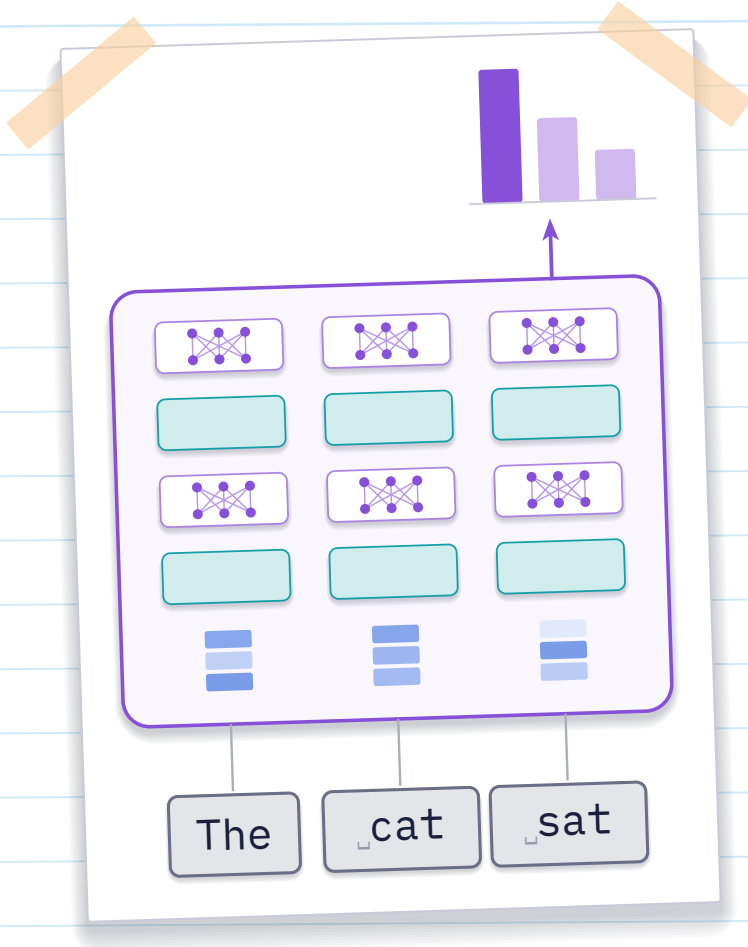
Prof. Aviv Zohar

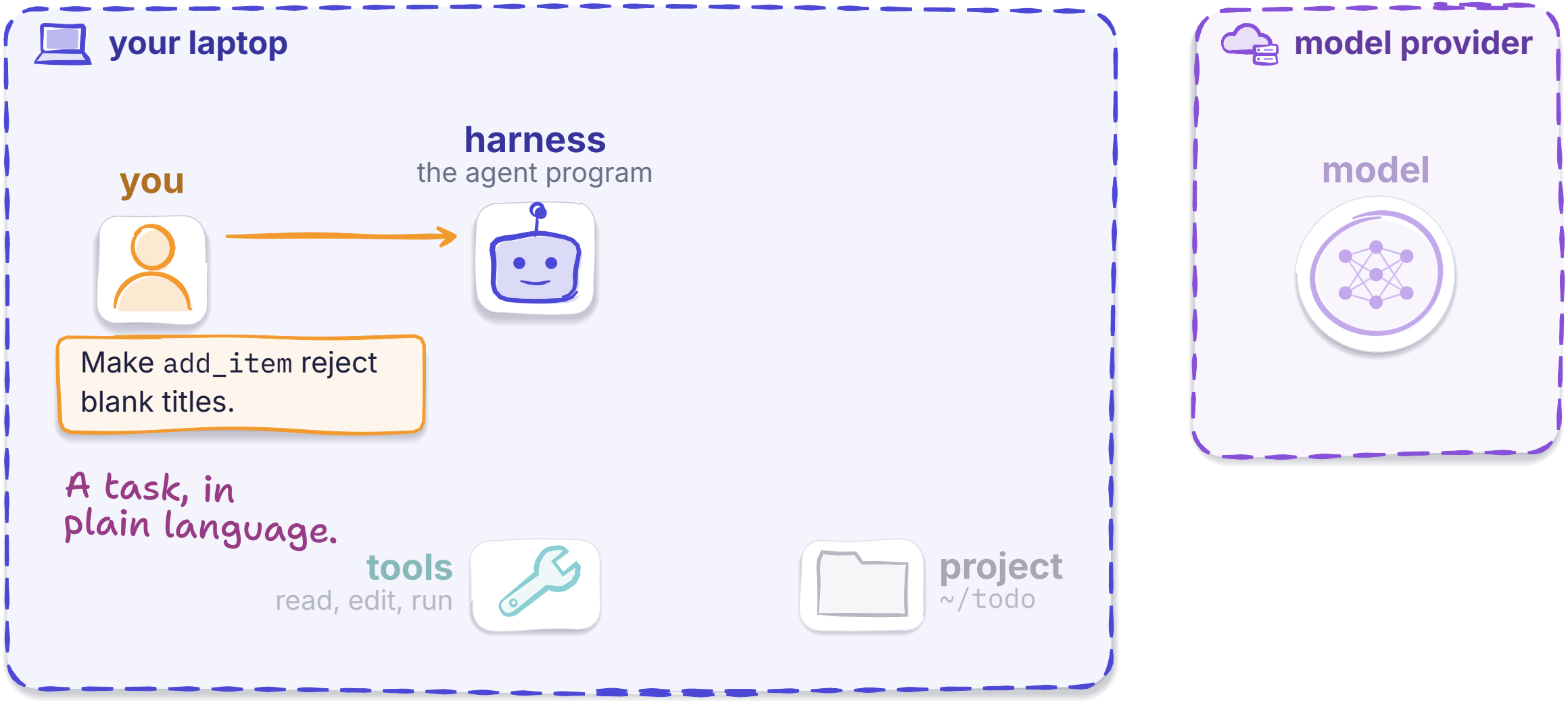
Semester A, 2026–7

Inside the Box: the Model's Role

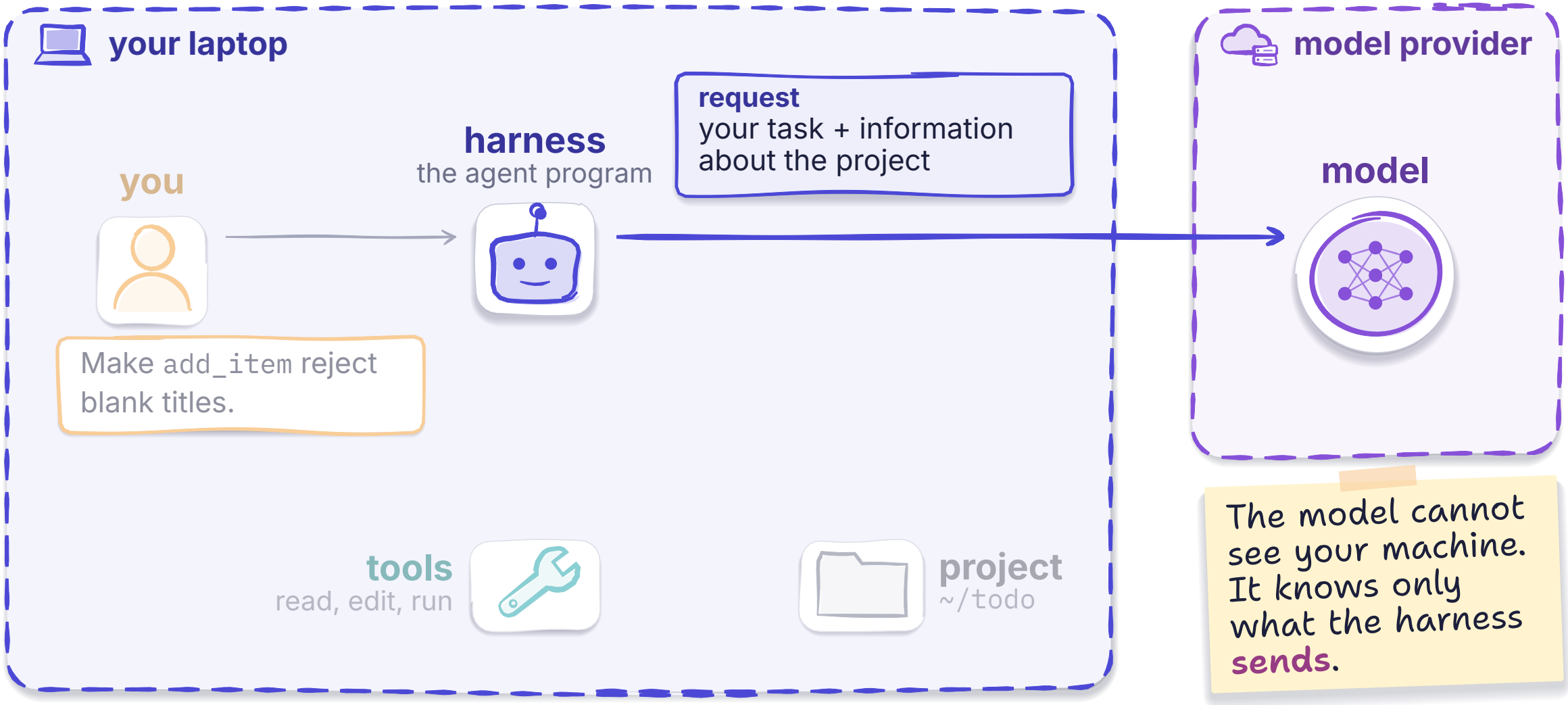
Tokens, embeddings, and next-token prediction



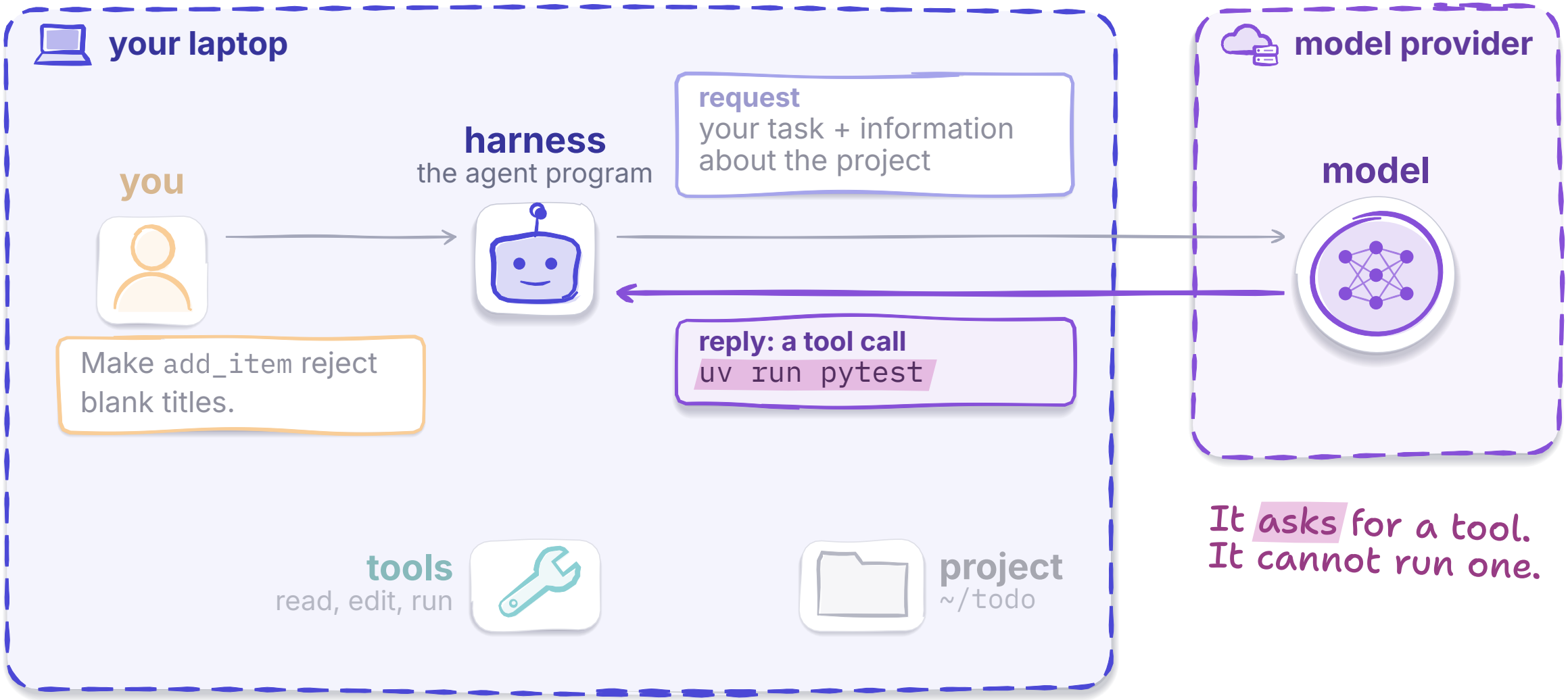
The harness runs the loop; the model proposes



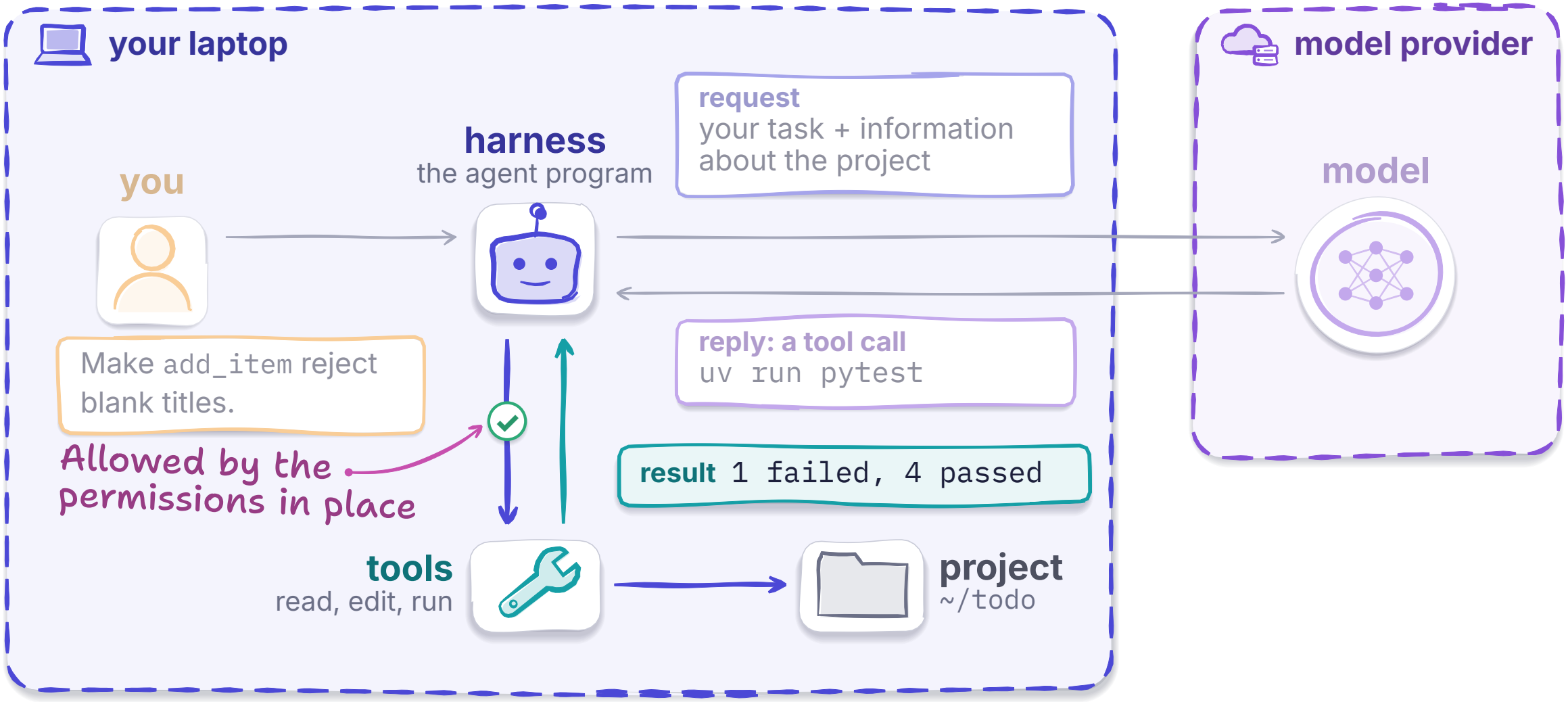
The harness runs the loop; the model proposes



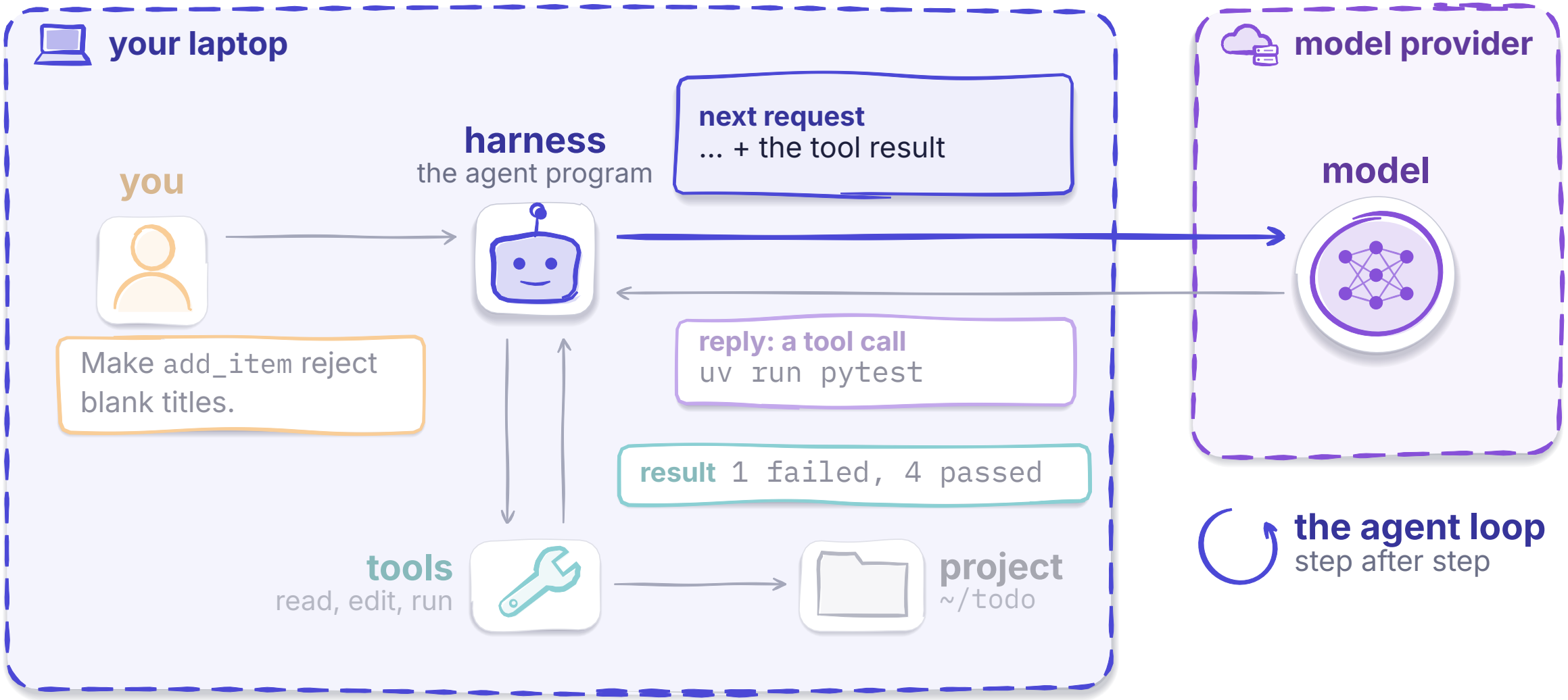
The harness runs the loop; the model proposes



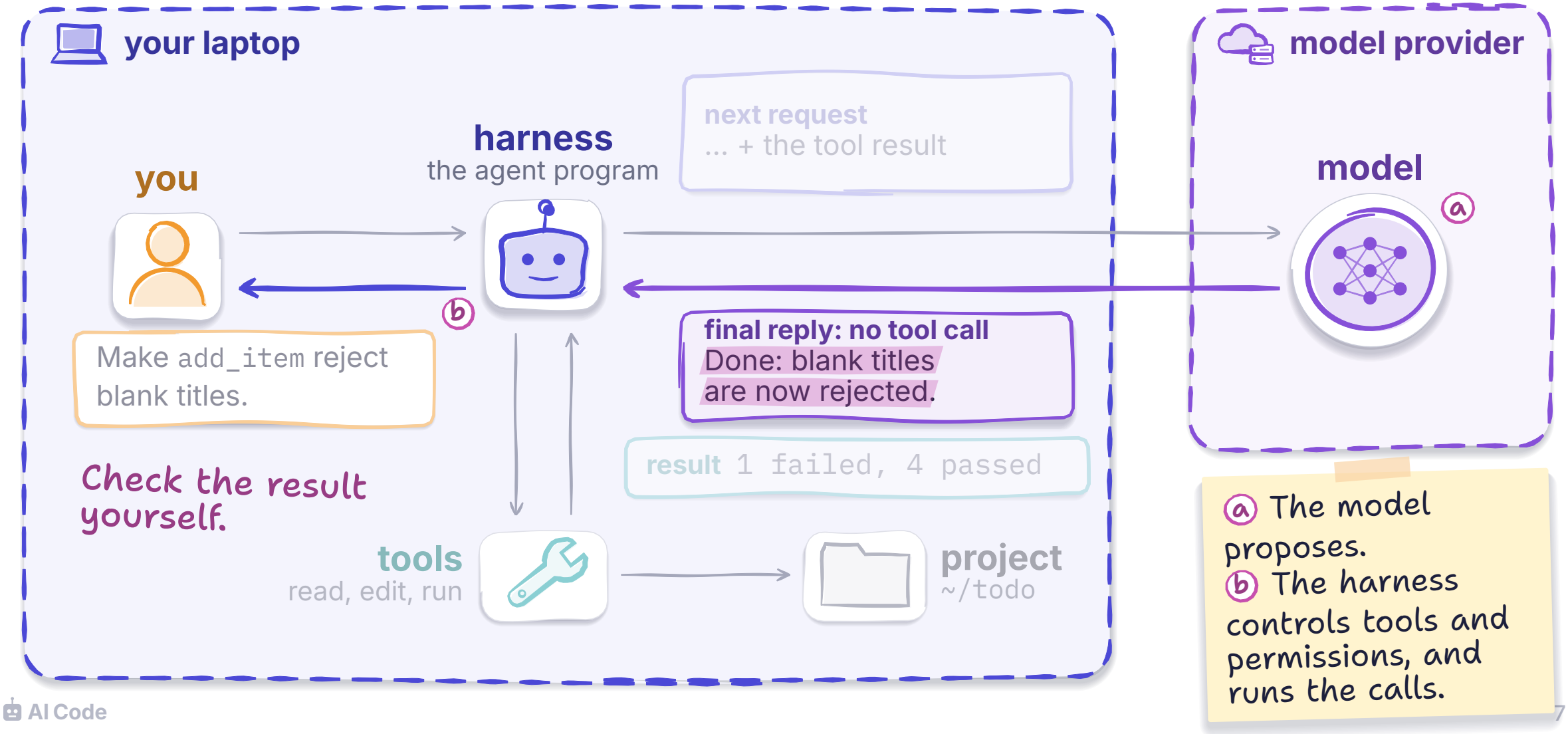
The harness runs the loop; the model proposes



The harness runs the loop; the model proposes



The harness runs the loop; the model proposes

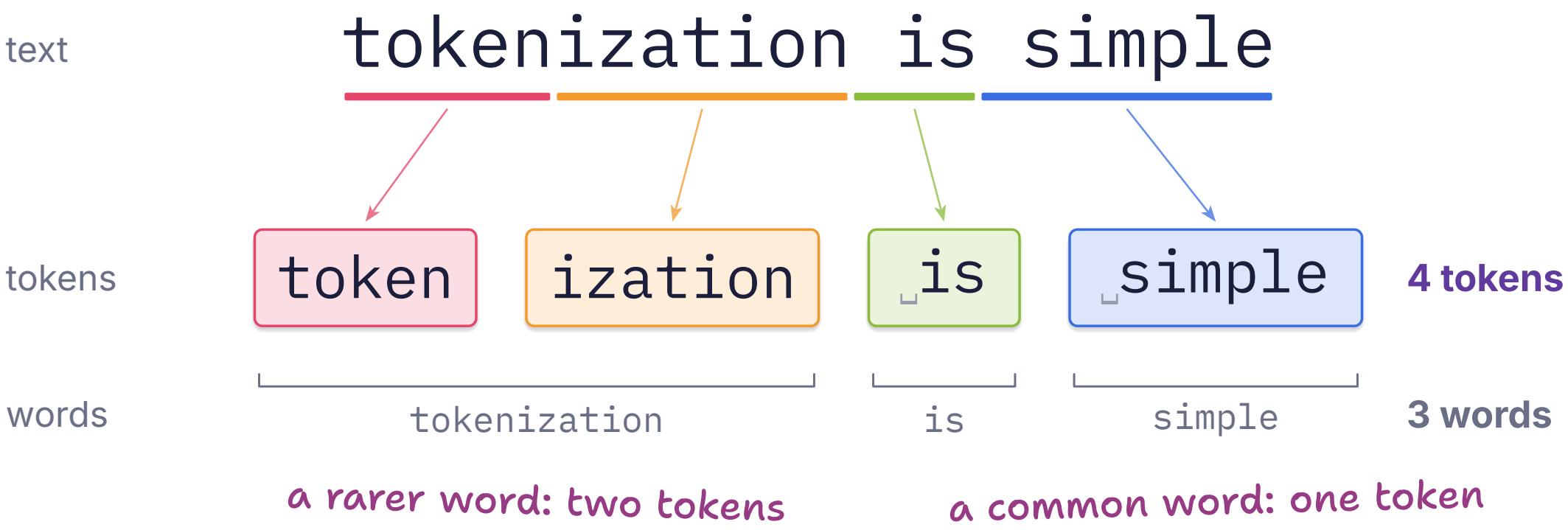


Text is split into tokens

text

```
tokenization is simple
```

Text is split into tokens



␣ *marks a space · the split depends on the tokenizer*

A tokenizer has a fixed vocabulary

≈128,000

tokens in Llama 3's vocabulary

an open model: its sizes are published



Llama 3 tokenizer: 128,256 tokens (Meta, 2024)

Counting letters is awkward for a model

How many **`r`**'s are in **`strawberry`**?

✗ 2

a familiar wrong answer

✓ 3

correct

Newer models often get this word right: it is in their training data now.

Counting letters is awkward for a model

you see



ten letters,
three r's

Counting letters is awkward for a model

you see



one token



One id for the whole word. No letters to count.

ids are illustrative · the split depends on the tokenizer

Counting letters is awkward for a model

you see



one token



a few tokens



Still chunks of letters, not letters.

ids are illustrative · the split depends on the tokenizer

Put the letters in the text, or count with code

spell it out



Each letter is now a token the model can count. It can still miscount.

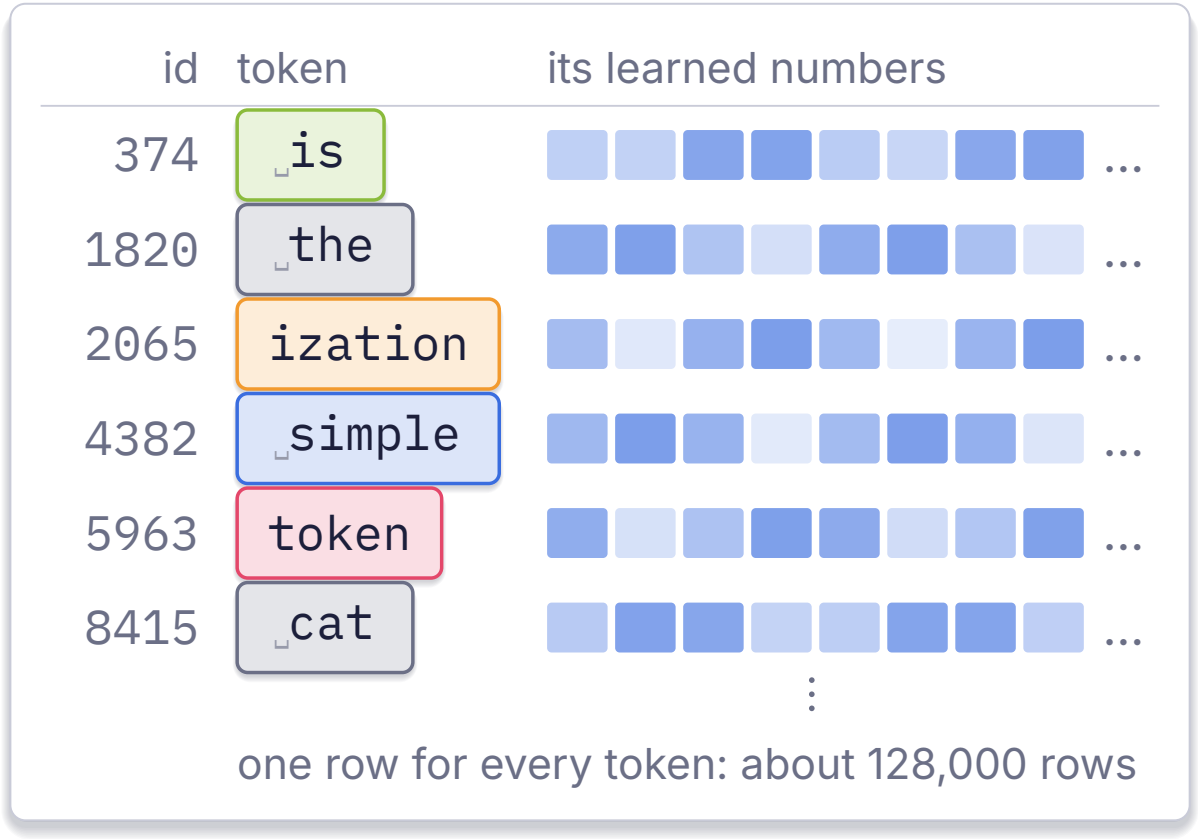
run code

```shell
~ $ python3 -c 'print("strawberry".count("r"))'
3
```

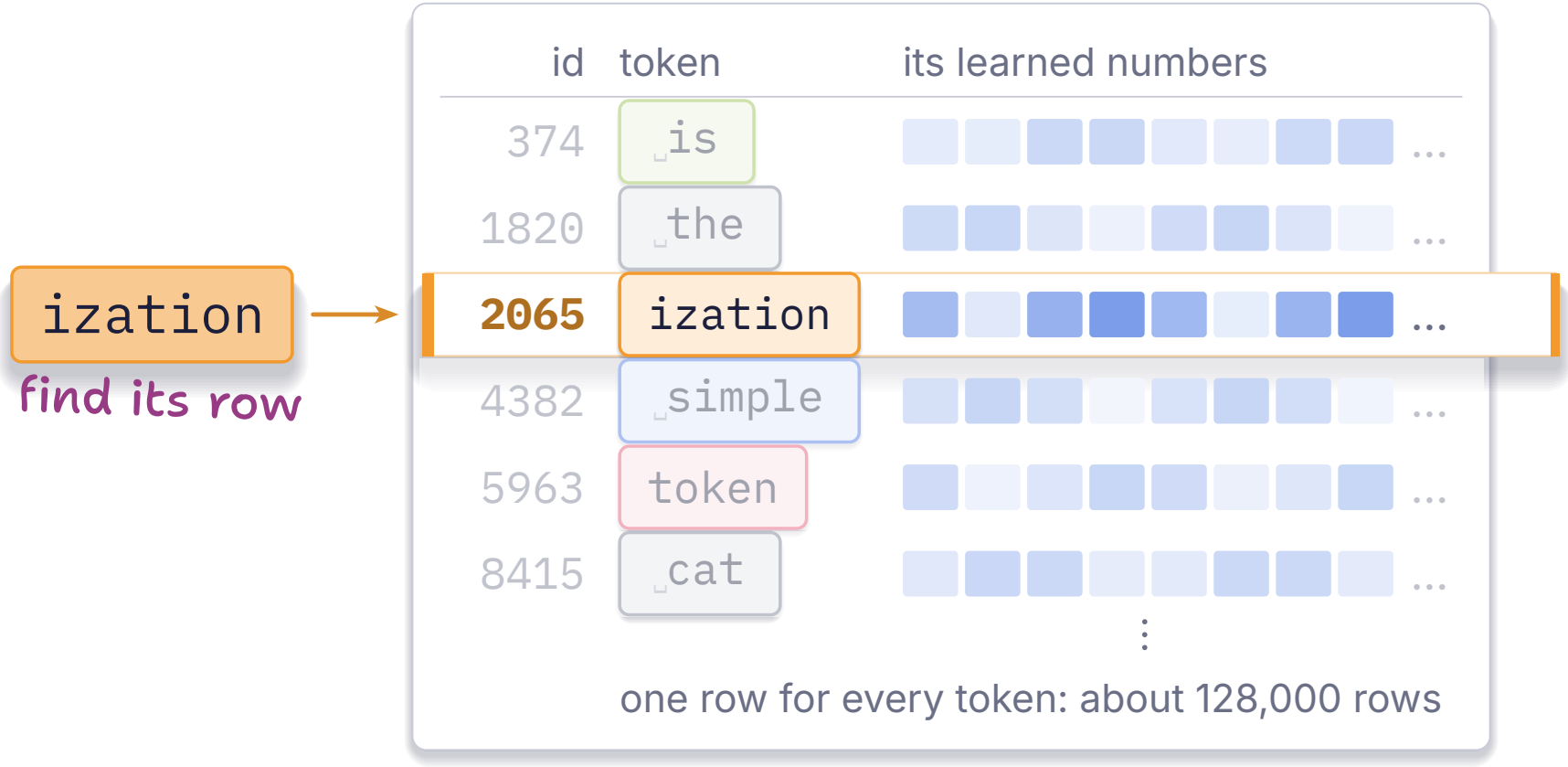
exact

A tool runs the exact operation, and we can read the code and check it.

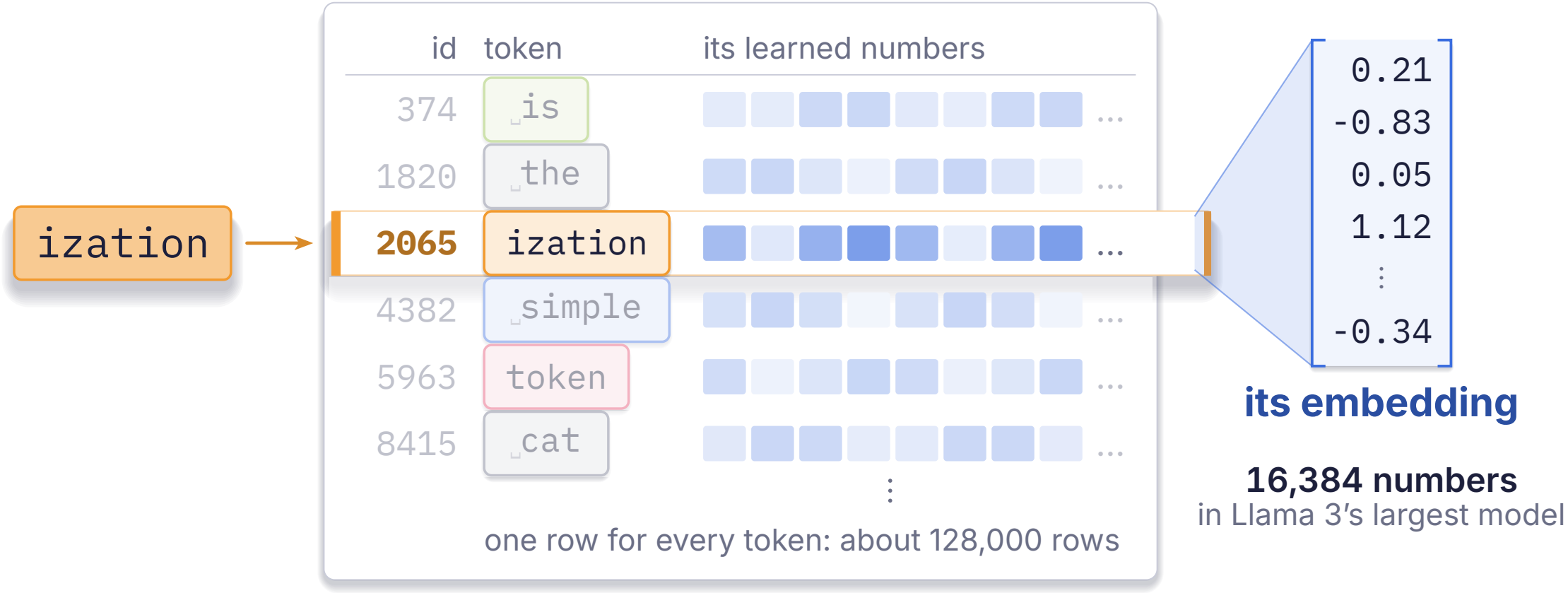
Each token looks up a learned vector



Each token looks up a learned vector

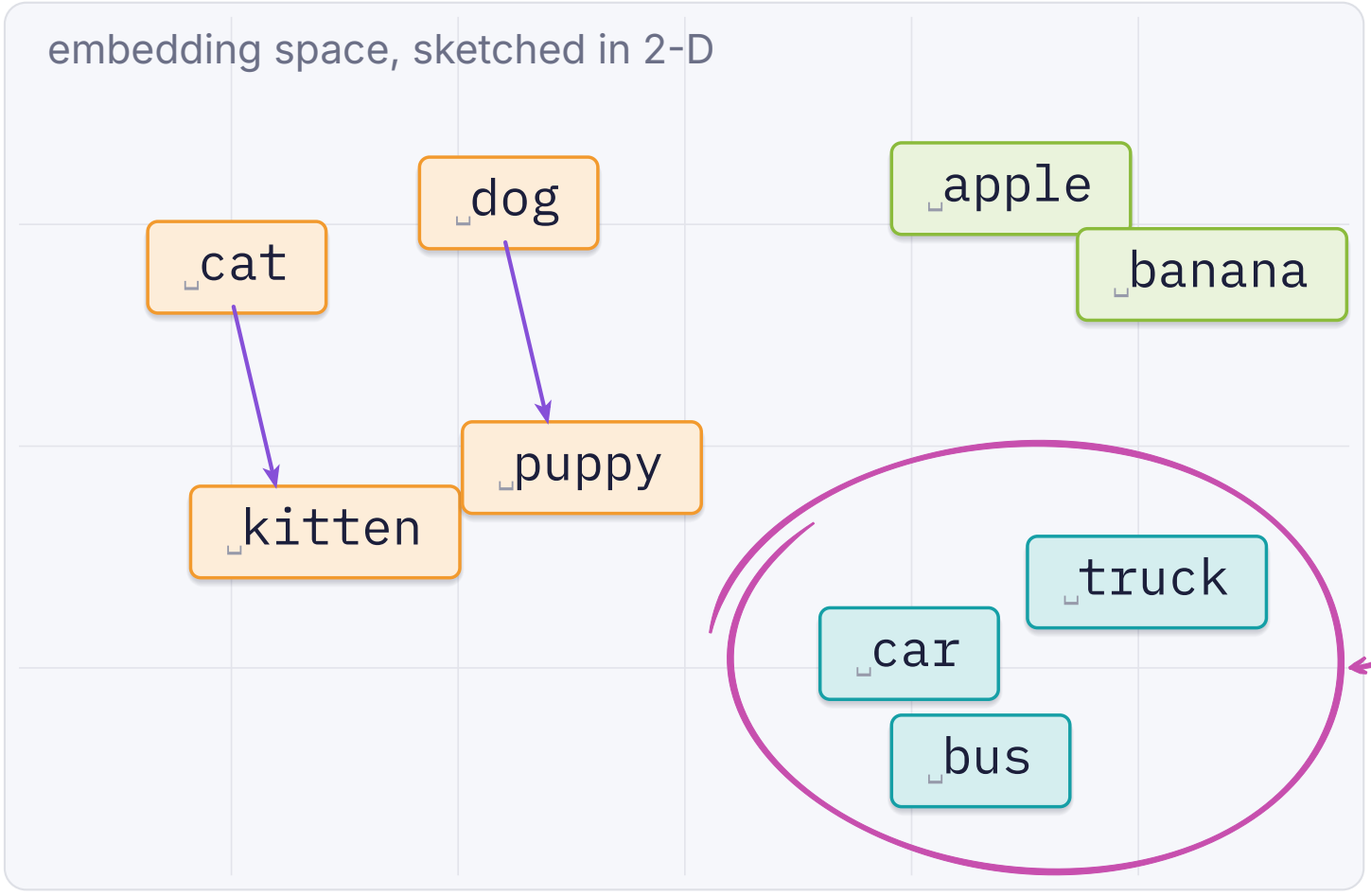


Each token looks up a learned vector



Llama 3 405B: 16,384 numbers per embedding · values illustrative

Training places related tokens near each other



The values start random. Training adjusts them.

The same direction: the same relation (cat → kitten).

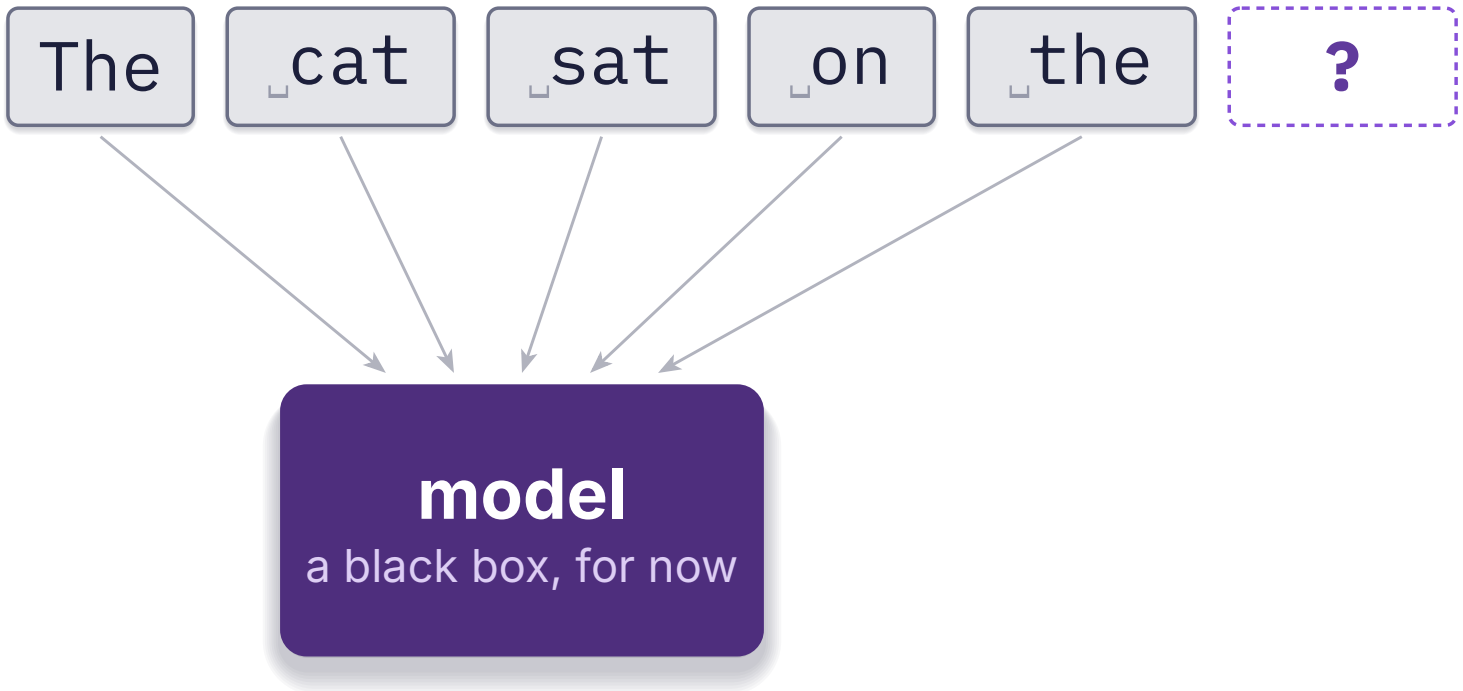
Similar tokens end up close.

a 2-D sketch: real embeddings have thousands of dimensions

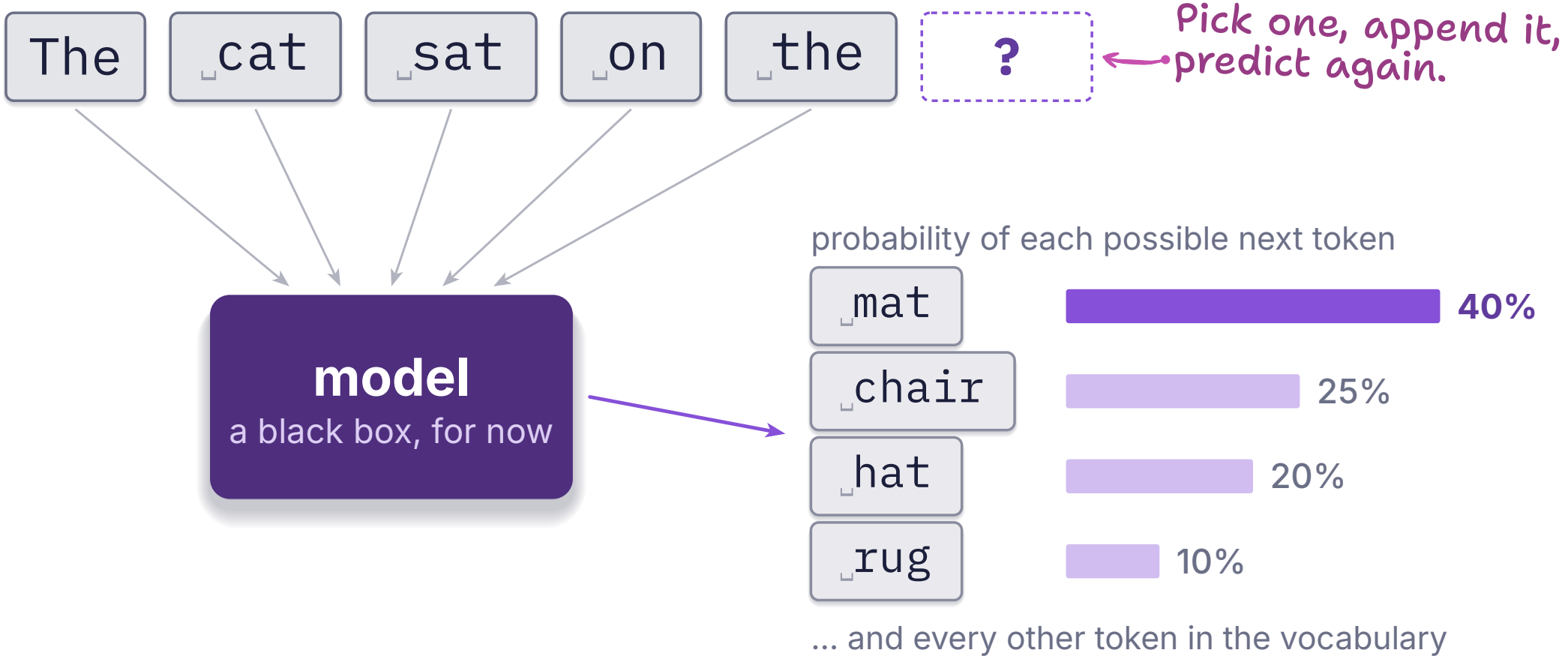
The model predicts the next token



The model predicts the next token



The model predicts the next token

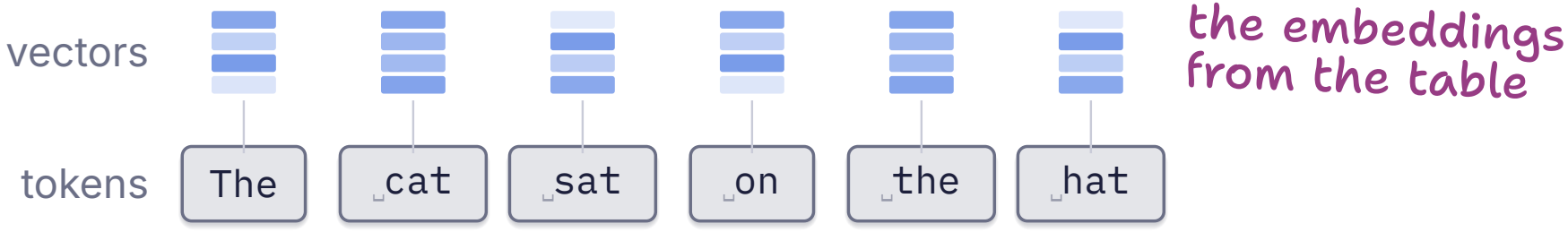


probabilities are illustrative

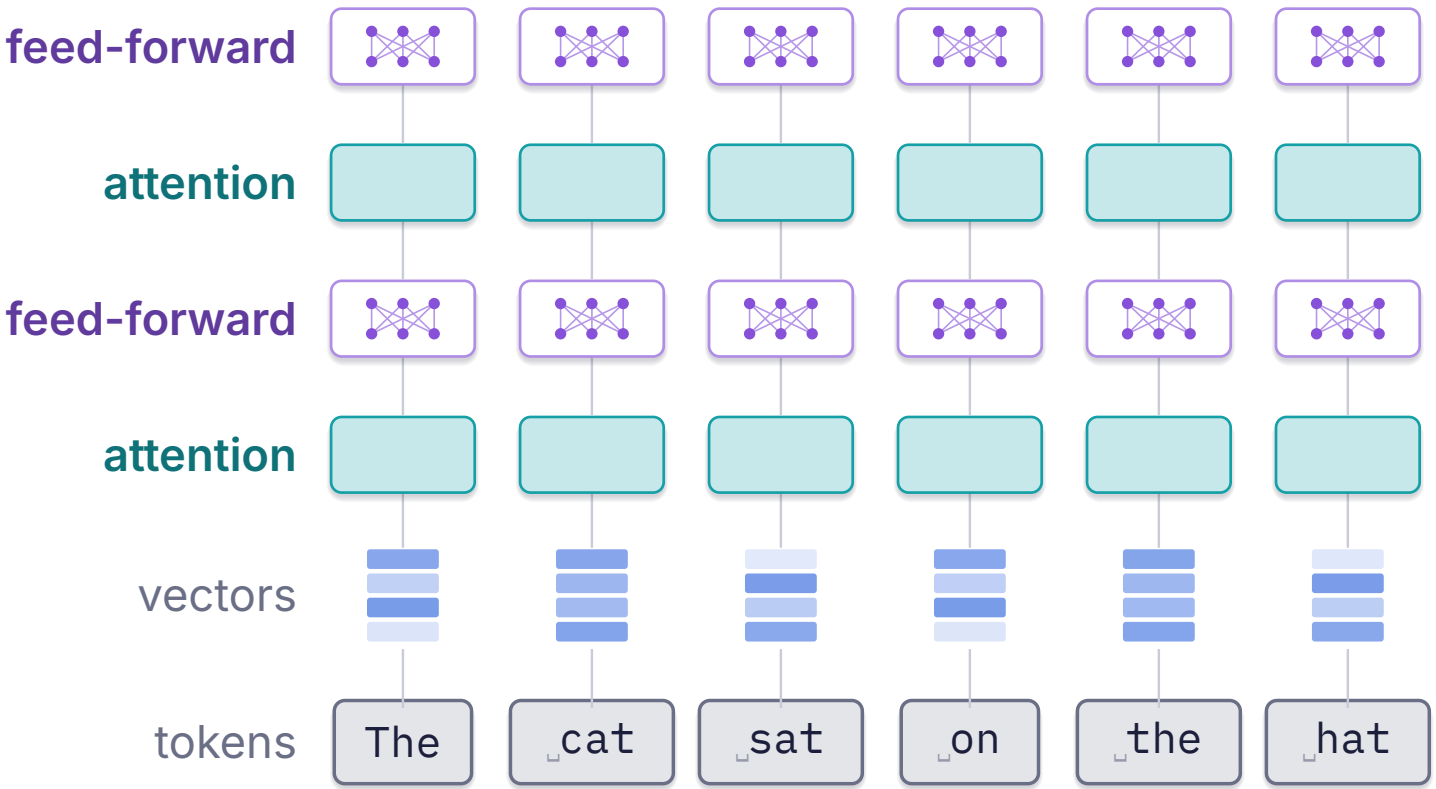
Inside the model: a stack of layers over each token



Inside the model: a stack of layers over each token

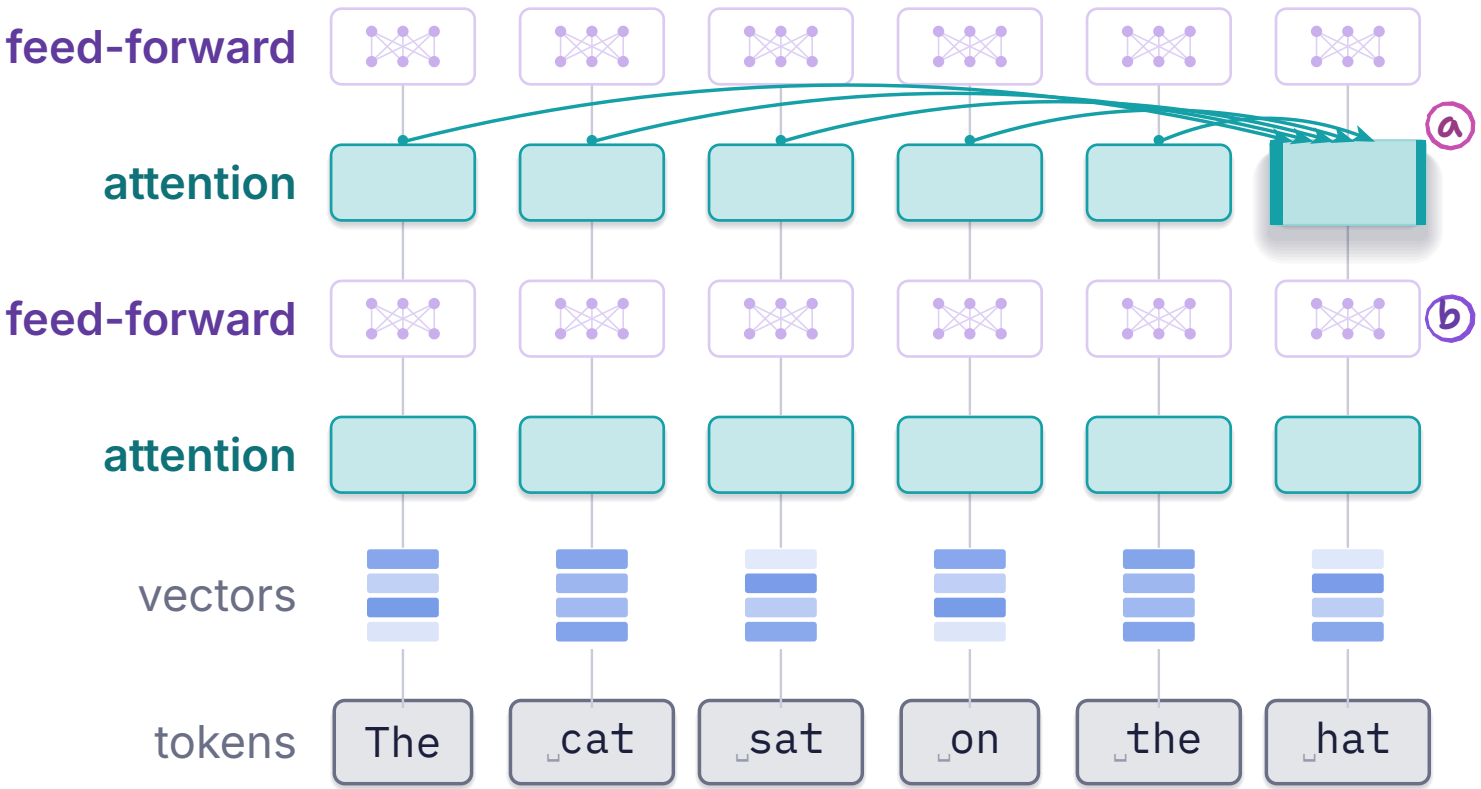


Inside the model: a stack of layers over each token



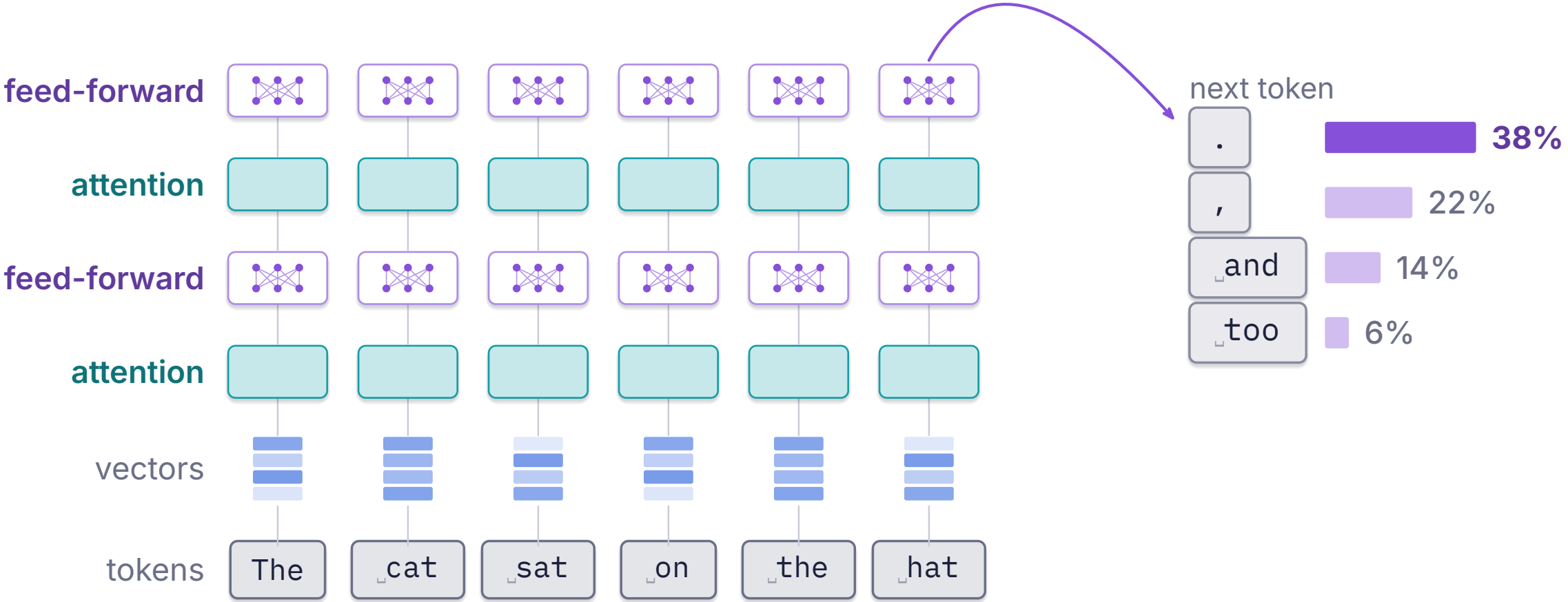
Real models stack many more layers. Here: two of each.

Inside the model: a stack of layers over each token



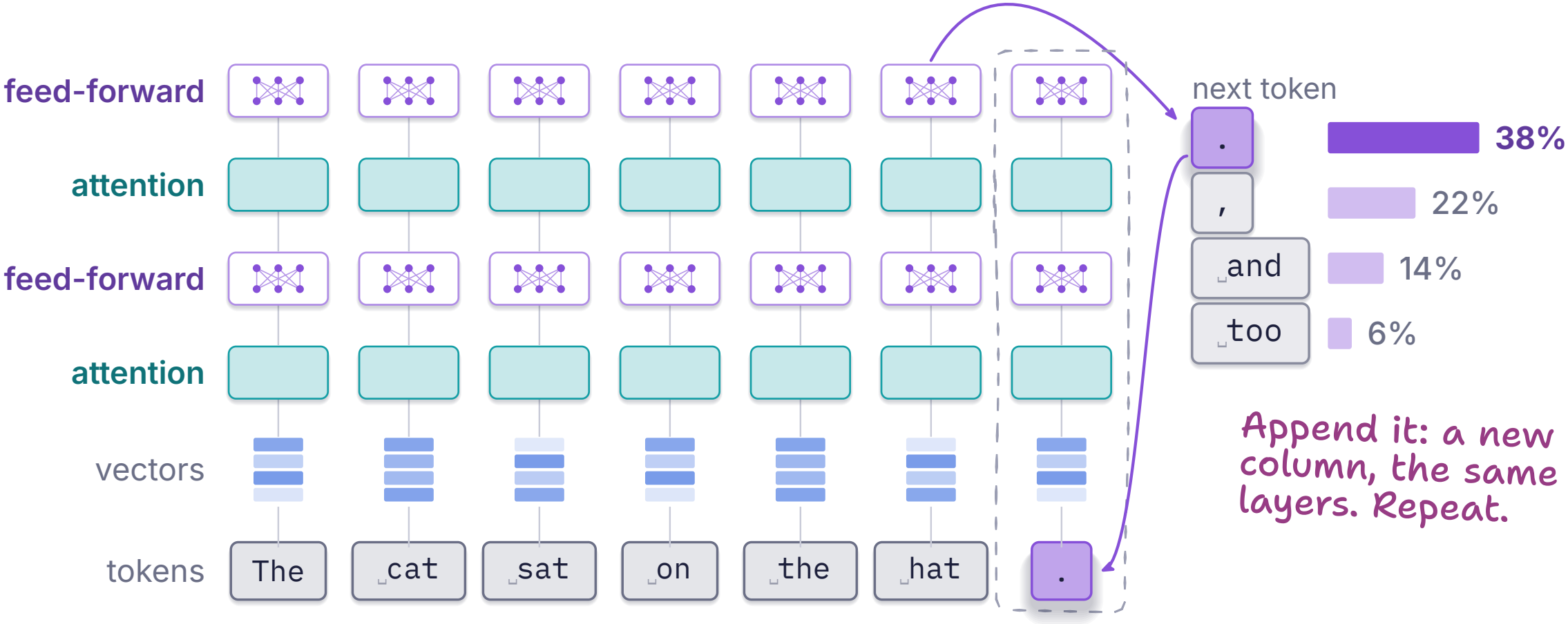
ⓐ **Attention:** a token pulls in information from the earlier tokens.
ⓑ Feed-forward works on each position alone.

Inside the model: a stack of layers over each token



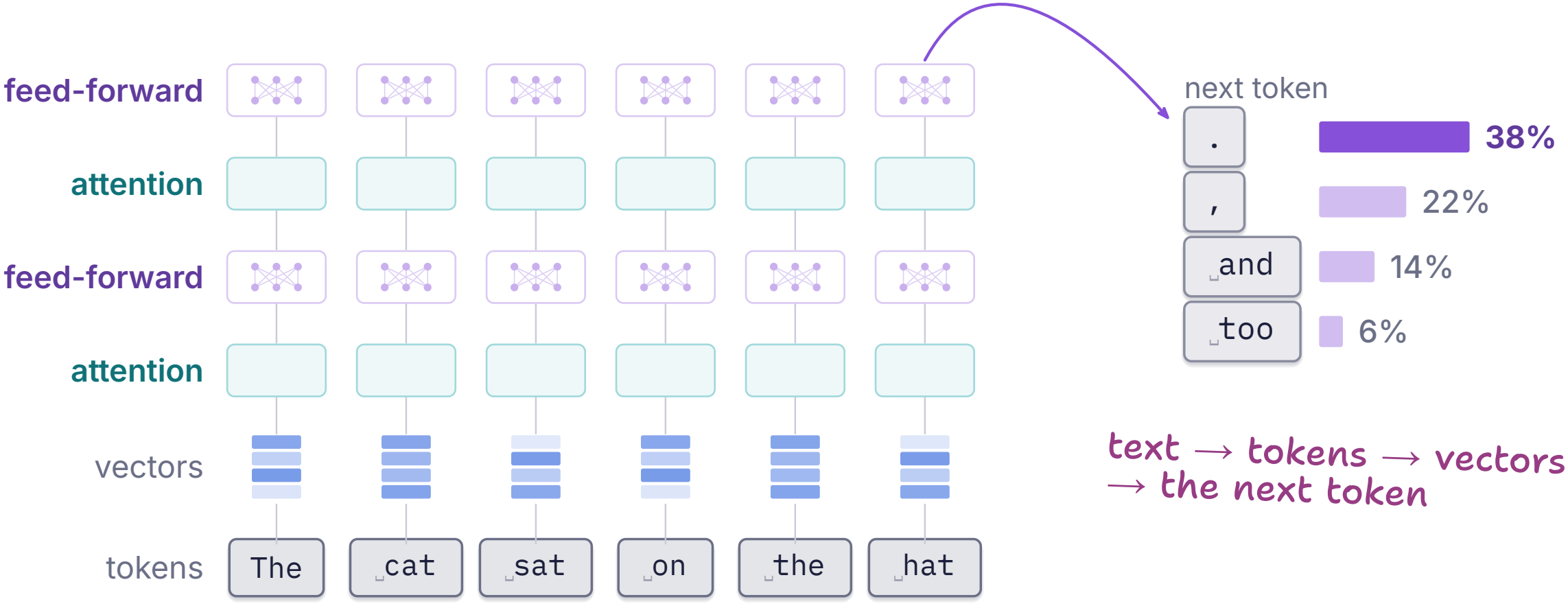
probabilities are illustrative

Inside the model: a stack of layers over each token

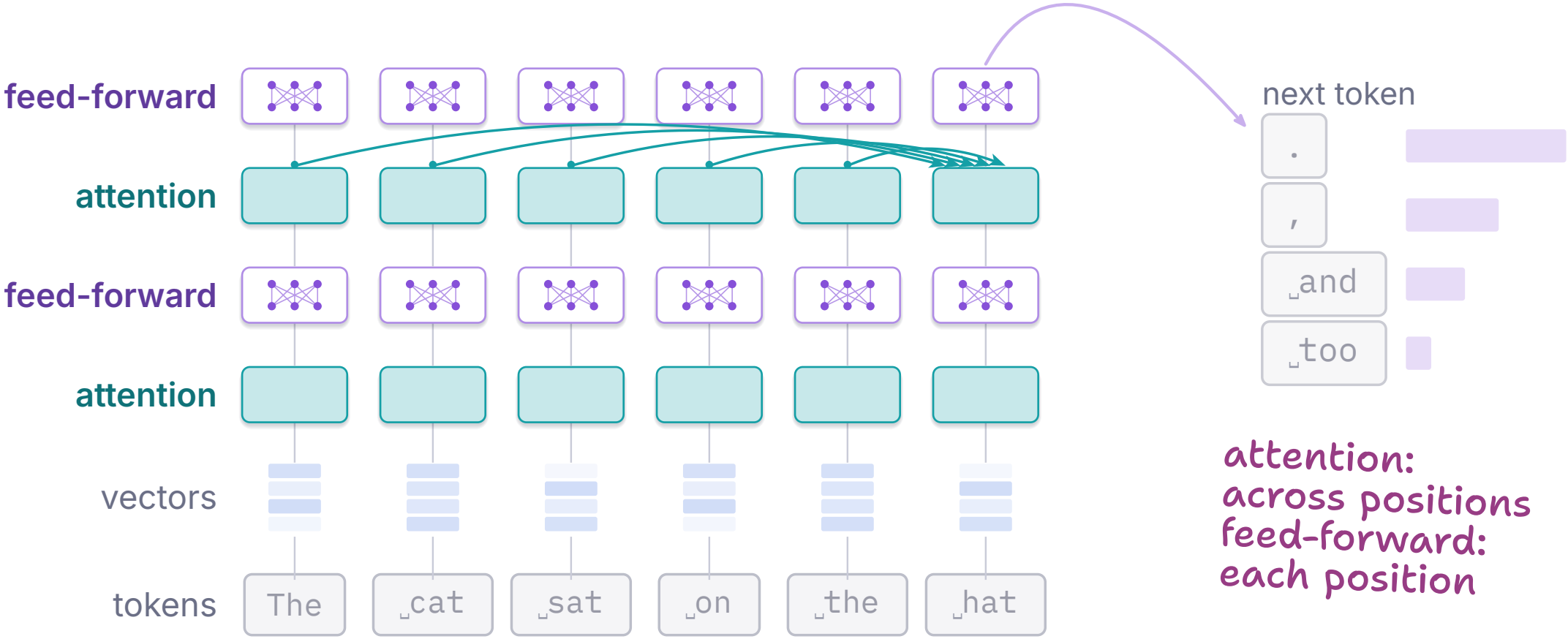


probabilities are illustrative

One predictor, from text to the next token



One predictor, from text to the next token



attention:
across positions
feed-forward:
each position

One predictor, from text to the next token

Every reply and every tool call comes from this one predictor.

