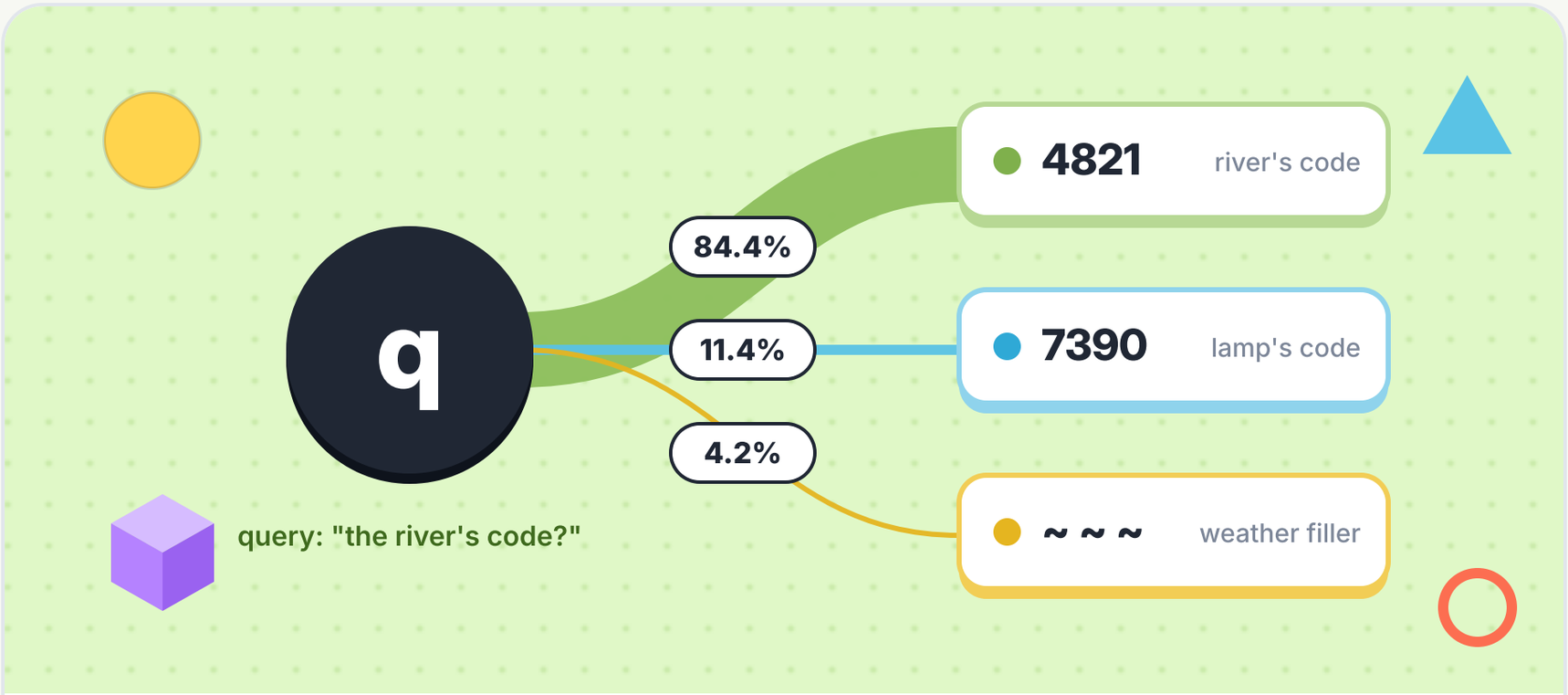


AI TUTORIAL · DAY 004 · Transformers

Attention, by hand

Work out where a transformer looks, using three tokens and a calculator.

7 steps · ~4 min · Only needs e^x

Start lesson

Mohammad Ashfaq Ur
AI Tutorials · Daily AI Notes

THE BIG PICTURE

The whole formula is three moves

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Vaswani et al., "Attention Is All You Need" (2017), Eq. 1

1

Score

Dot product of the query with every key, divided by $\sqrt{d_k}$. One number per token: how well it matches.

$q \cdot k \div \sqrt{d_k}$

2

Normalise

Softmax: raise e to each score and divide by the total. Weights are positive and add up to 1.

[3, 1, 0] → [.84, .11, .04]

3

Mix

Multiply each token's value vector by its weight and add them up. That sum is the output.

$\Sigma\, w_i \cdot v_i$

Q **query**: what the current token is looking for

K **key**: what each token offers to be matched on

V **value**: what each token passes on if picked

Continue

STEP 1 · SCORE

Score every token against the query

What is the secret code of the river? → QUERY VECTOR **q = [2, 0, 1, 1]**

TOKEN	KEY K	Q · K	÷ √4	SCORE
river's code is 4821	[2, 0, 1, 1]	6	÷ 2	**3**
lamp's code is 7390	[0, 2, 1, 1]	2	÷ 2	**1**
the weather was mild	[0, 1, 0, 0]	0	÷ 2	**0**

WORKED: THE RIVER FACT

$2\times2 + 0\times0 + 1\times1 + 1\times1 = \mathbf{6}$

$6 \div \sqrt{4} = 6 \div 2 = \mathbf{3}$

WHY DIVIDE BY $\sqrt{d_k}$?

Dot products grow with vector size. Large scores push softmax into a region where gradients get tiny (Vaswani et al., §3.2.1).

Toy setup: each statement squeezed into one token · $d_k = 4$ · one head · no positions or learned projections.

Continue

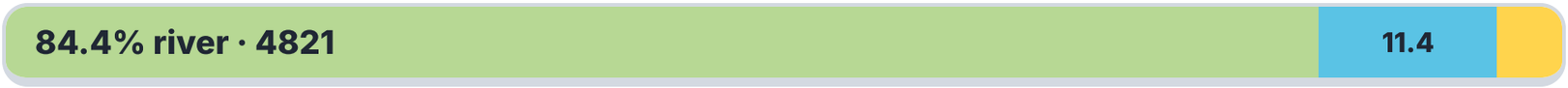
STEP 2 · SOFTMAX

Turn scores into a 100% budget

TOKEN	SCORE	E^SCORE (ITS PULL)	SHARE
river's code is 4821	3	20.09	84.4%
lamp's code is 7390	1	2.72	11.4%
the weather was mild	0	1.00	4.2%

total = 20.09 + 2.72 + 1.00 = 23.80 · share = e^score ÷ 23.80

WHERE THE ATTENTION GOES



Every token gets a slice; the slices always add up to 100%

4.2% weather ↗

84.4%

ON THE EVIDENCE

of the attention budget goes to the river fact

RULE OF THUMB

+1 score = ×2.72

pull = e^score, and e ≈ 2.718

Continue

Mohammad Ashfaq Ur
AI Tutorials · Daily AI Notes

STEP 3 · MIX

Blend the values using those weights

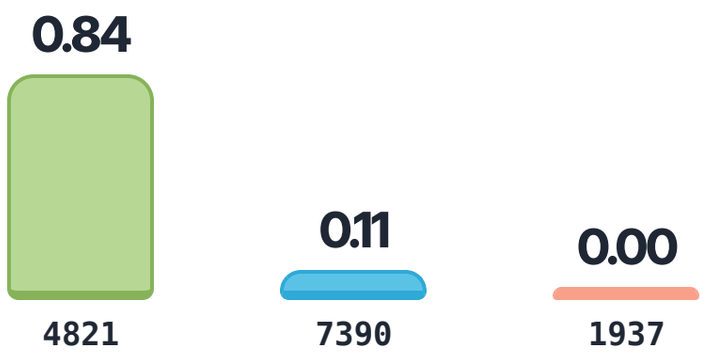
TOKEN	WEIGHT	VALUE V
river's code is 4821	0.844	[1, 0, 0]
lamp's code is 7390	0.114	[0, 1, 0]
the weather was mild	0.042	[0, 0, 0]

Value slots = answers [4821, 7390, 1937]. The weather line carries no answer.

THE OUTPUT

$$0.844 \cdot [1,0,0] + 0.114 \cdot [0,1,0] + 0.042 \cdot [0,0,0] = [0.84, 0.11, 0.00]$$

OUTPUT, BY ANSWER



✓ The model leans hard towards 4821.

Continue

TWIST · ADD A RIVAL

One lookalike cuts the evidence to 46%

NEW TOKEN · river's code is 1937 · `k = [2, 0, 1, 1] → score 3, same as the real fact`

BEFORE



AFTER



Output is now **`[0.46, 0.06, 0.46]`**: 4821 and 1937 are tied. A coin flip.

QUICK CHECK

Filler tokens score 0. How many would it take to pull as much attention as one rival scoring 3?

- A 1
- B 3
- C about 20
- D about 400

Check

Answer on the next slide →

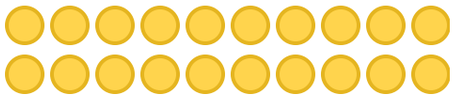
Mohammad Ashfaq Ur
AI Tutorials · Daily AI Notes

Answer: C, about 20

A token's pull is e^score, so one rival = $e^3 \div e^0$ = **20.1** filler tokens.

SAME PULL

1 rival =



Bonus: if the model scores filler at −3, you'd need $e^6 \approx$ **403** of them.

TAKEAWAY

Attention is a fixed budget split by e^score. Whatever looks most like the answer takes the biggest share, right or wrong.

CHECK IT IN CODE

```
import torch
scores = torch.tensor([3., 1., 0.])
torch.softmax(scores, dim=0)
# tensor([0.8438, 0.1142, 0.0420])
```

WHERE THIS SHOWS UP · DAY 003

Liu (2026): attention left on the evidence explained up to **58%** of the variance in retrieval loss. Log length explained at most **8%**.

REFERENCES

arxiv.org/abs/1706.03762
pytorch.org/docs · scaled_dot_product_attention
arxiv.org/abs/2611.04169



Mohammad Ashfaq Ur
Follow for daily AI notes and tutorials
ashfaqfardin.dev