

Season 1: How LLMs work

Attention.

How a model decides which earlier tokens matter for the next one

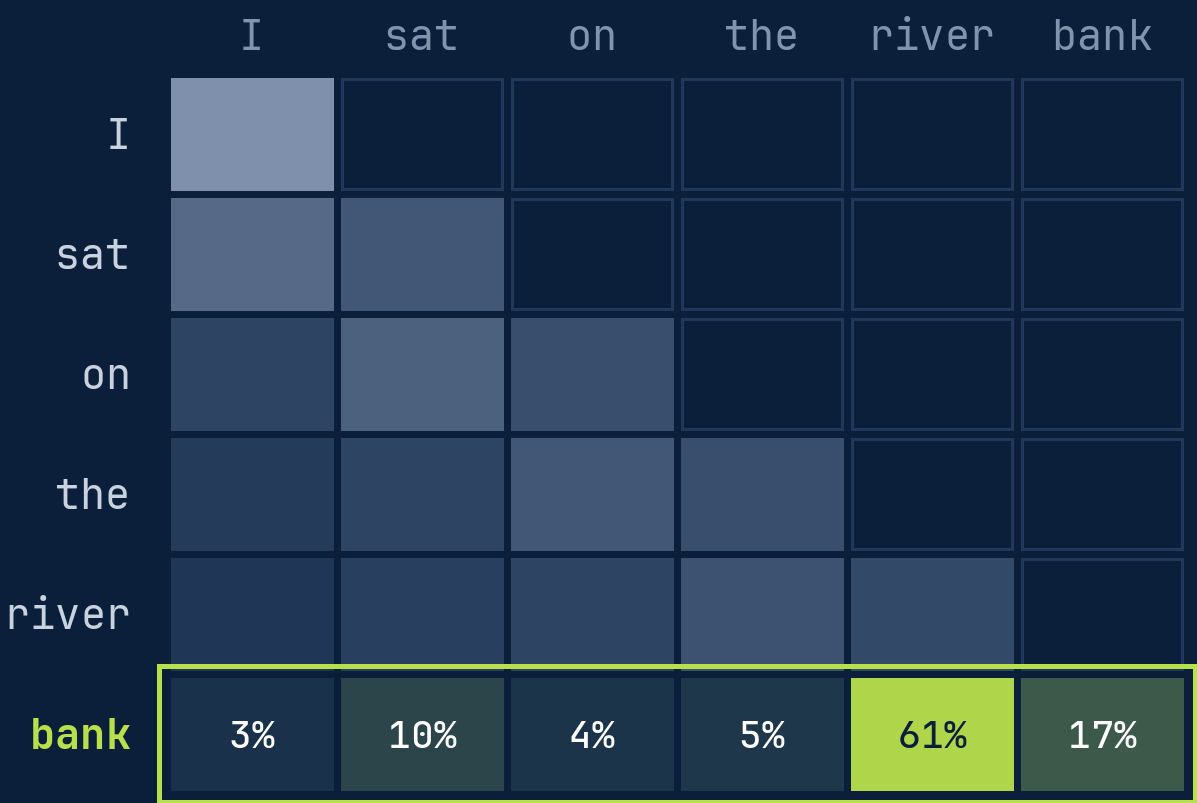
Why context works at all

- **Meaning depends on context:** "bank" means one thing beside "river" and another beside "loan"
- **Long prompts:** attention is how a model reaches back to a fact thousands of tokens earlier (S1 E7)
- **Cost:** each token is compared with every token before it, so long prompts cost more to process

Every token looks back at every token before it

Each token scores how relevant each earlier token is, then blends in their information by those scores.

WHO CAN SEE WHOM "I sat on the river bank"



Read across a row
Each row is one token, scoring the tokens before it and itself

Dark cells
Later tokens are not written yet, so they are never visible

The bank row
Most weight goes to river, so the model reads a riverbank, not a lender

Illustrative weights for one head; each row sums to 100%. The filled triangle is also why cost grows with the square of the length (slide 7).

Where it breaks

Attention shows where the model looked, not why it answered. It is not a reliable explanation of a decision.

Query, key, and value

Three vectors, computed from each token's embedding (S1 E3).

IN A LIBRARY

Query	What this token is looking for	your search
Key	What each visible token offers	the spine label
Value	The information passed on	the book itself

WORKED EXAMPLE

What should "bank" pay attention to?

KEY	1 SCORE	2 WEIGHT (softmax: shares of 100%)
river	3.5	61%
bank (itself)	2.2	17%
sat	1.7	10%
the	1.0	5%
on	0.8	4%
I	0.5	3%

3 BLEND

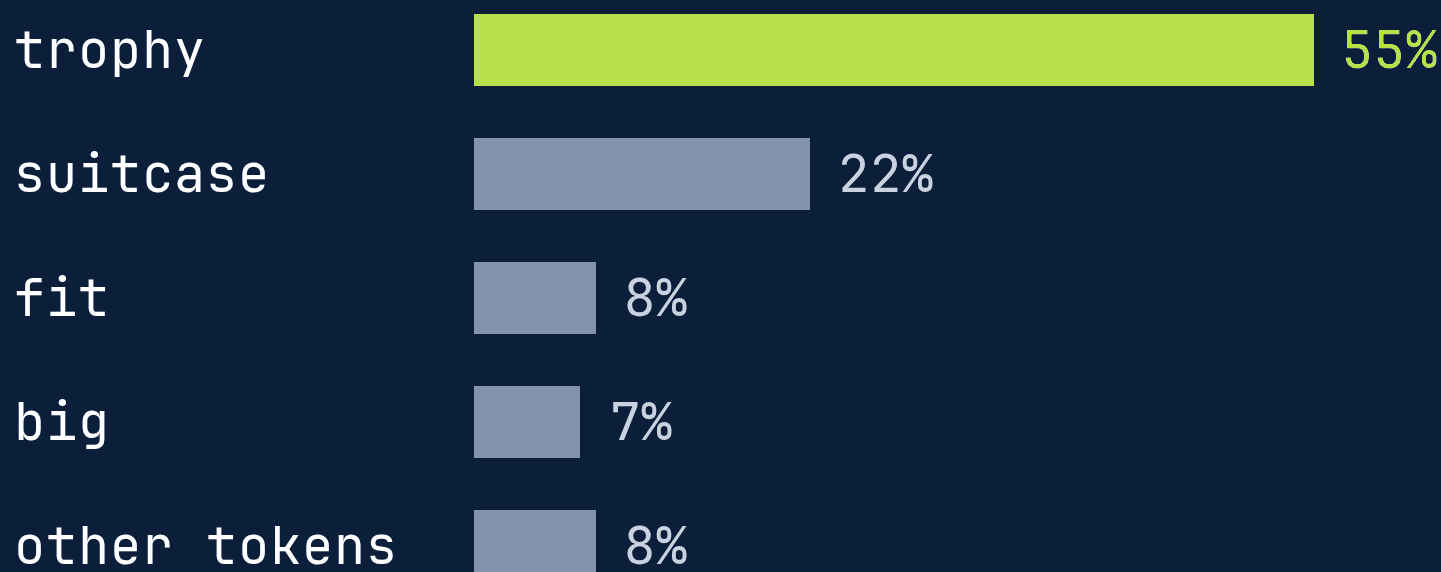
New vector for bank = 0.61 × river + 0.17 × bank + 0.10 × sat + the rest. It now carries mostly river meaning, so later layers read a riverbank, not a lender.

In one line

Score each key against the query, turn the scores into weights that sum to 1, and add up the values by those weights.

What "it" pays attention to

"The trophy did not fit in the suitcase because it was too big."



■ The strongest weight for the token "it"

"it" draws most of its meaning from "trophy", so the next tokens treat "it" as the trophy. Change "big" to "small", and the weight should shift to "suitcase".

Illustrative weights for one attention head. Real models spread this across many heads.

Many heads, many layers

HEADS side by side, each asking its own question

Each head has its own query, key and value, so "bank" collects several links at once.

	I	sat	on	the	river	bank
Meaning what kind of bank?	3%	8%	4%	3%	70%	12%
Grammar which word governs it?	5%	20%	55%	10%	4%	6%
Subject who is there?	60%	20%	3%	2%	5%	10%

The heads' outputs are joined, so the next layer sees all three links. Illustrative: real heads are learned, not labelled, and many have no neat name.

LAYERS stacked, each building on the last

↑	Late layers	"the riverbank where I sat"
	Middle layers	"a riverbank, not a lender"
	Layer 1	"bank": the word, meaning still open

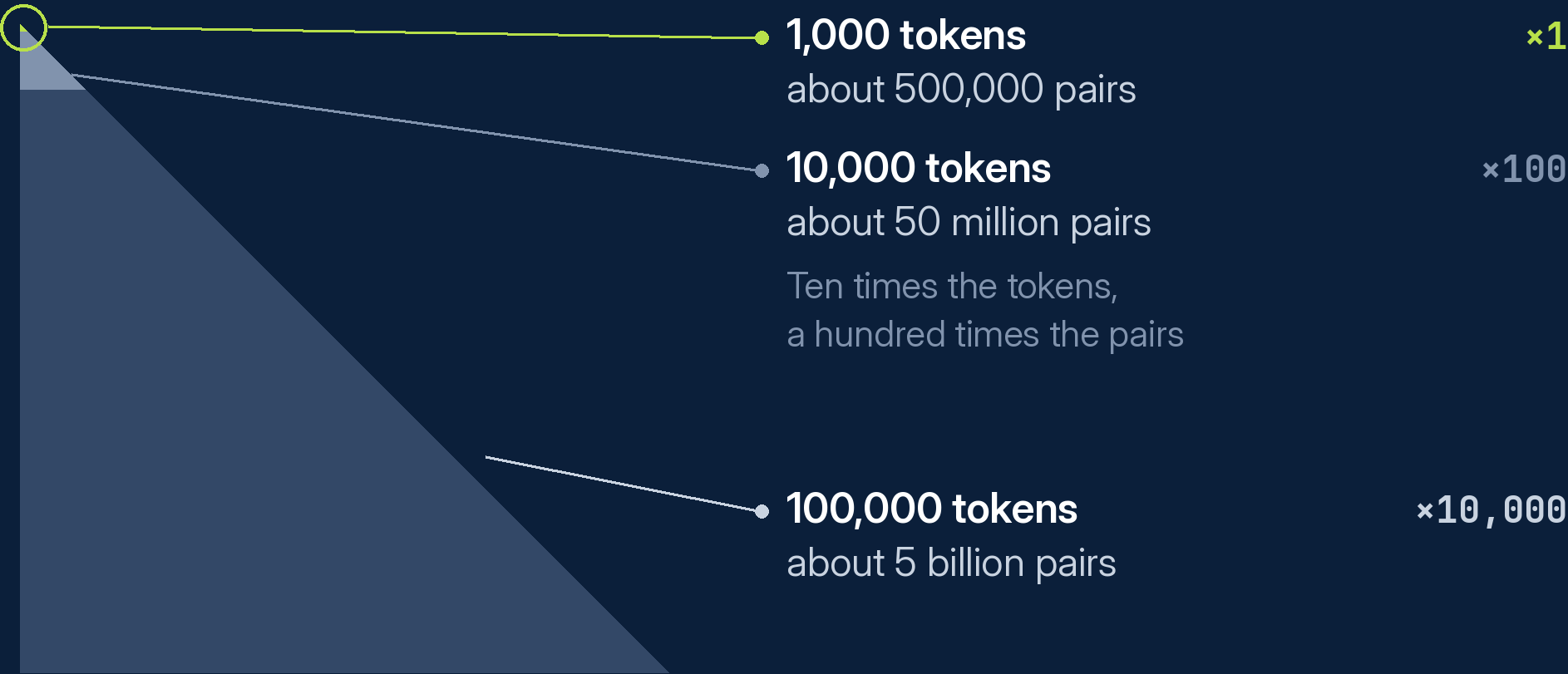
Each layer runs attention again on what the layer below produced, so links combine. Big models repeat this dozens of times.

Looking back only	A token can attend to earlier tokens, never later ones. This matches generation, where each new token is predicted from those before it (S1 E1)
--------------------------	---

S1 E5 puts these pieces together into the transformer.

Ten times the context, a hundred times the comparisons

Every token is compared with every earlier token, so the work grows with the square of the length.



Drawn to scale: each triangle's area is the number of token pairs, per layer and per head. The speck circled in lime is 1,000 tokens.

Speed	Time to first token grows with prompt length. Test at the length you will actually send.
Price	Input is usually billed per token, so the bill looks linear. The square shows up in speed and in higher long-context rates.
Workload	At short lengths the rest of the model does most of the work. Attention takes over as prompts get long.

More text means thinner attention

- **Competition:** every token in the prompt competes for the same weights, including the irrelevant ones
- **Look-alikes:** passages that resemble the answer can draw attention away from it
- **Simple is not hard:** S1 E7 found no position effect for one distinctive fact up to 64,000 tokens. Harder tasks were not tested
- **Heatmaps mislead:** a weight shows where the model looked, not what it concluded

What this means for prompts

Send what matters	Retrieve the relevant passages rather than pasting everything you have (Season 2)
Keep the opening fixed	Providers cache attention work for a repeated prompt opening. In S1 E7, Gemini served 25% of its input from cache
Measure with length	Time to the first token rises with prompt size. Test latency at the length you will actually send

What breaks in production

- **Stuffing the window:** sending everything because it fits, and paying for noise on every call
- **Long context as memory:** a big window is capacity, not reliable recall of everything in it
- **Breaking the cache:** putting changing content first, so the repeated opening is never reused
- **Heatmaps as proof:** using attention pictures to explain or defend a model's decision

A bigger context window is capacity, not focus

Every token sent is processed on every call, and more text gives the model more ways to be distracted.

Ask your team

How much of what we send on each call is actually needed, and have we tested at the full length we plan to use?

Decision rule

Send the model what it needs, not everything you have: every extra token competes for attention and adds cost.

Next: S1 E5 The transformer

How embeddings, attention, and layers fit together into one model.

Where this fits

The series map

S1 LLMs	S2 Retrieval	S3 Strategy
S4 Adapting	S5 Agents	S6 Economics
S7 Data	S8 Evaluation	S9 AI teams
S10 Reliability	S11 Security	S12 Delivery
S13 Performance	S14 Resilience	S15 Governance

☐ Leadership season

 Follow	linkedin.com/in/brendan-kelly-irl
 Episode guide	brendankellyai.github.io
 Code	github.com/BrendanKellyAI/applied-ai-lab