Intro to Transformers

Catherine Ning

Deep Learning Part I, Module 4 of 15.S60 IAP Software Tools 2026

What is “GPT”?

GPT = **Generative Pre-Trained Transformer**

- **generative** means the model generates new text
- **pre-trained** means the model was trained on large amounts of data
- **Transformer** = special kind of neural network used for text-to-text, voice-to-text, text-to-voice, text-to-image, machine translation, etc.

3 common **Transformer** families:

- Encoder-only (BERT-like): representations for classification/retrieval
- Decoder-only (GPT-like): generate text / predict next word
- Encoder--decoder (T5-like): seq2seq tasks (translation, summarization)

What a GPT model does

A GPT is trained to predict what comes next: a probability distribution over tokens.

Generation loop (sampling)

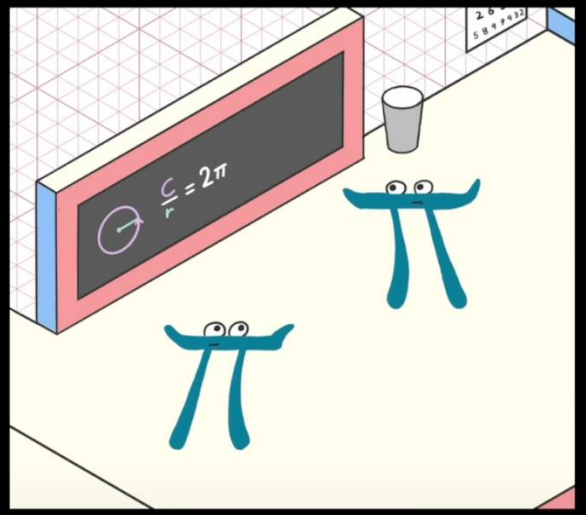
1) Feed prompt → distribution over next token
2) Sample (or pick max)
3) Append token and repeat

This simple loop turns “predict next token” into “generate long text.”

Key idea

The model is a learned function with many parameters (weights).
At inference time, the weights are fixed; only the data flowing through changes with the prompt.

Behold, a wild pi creature,
foraging in its native _____

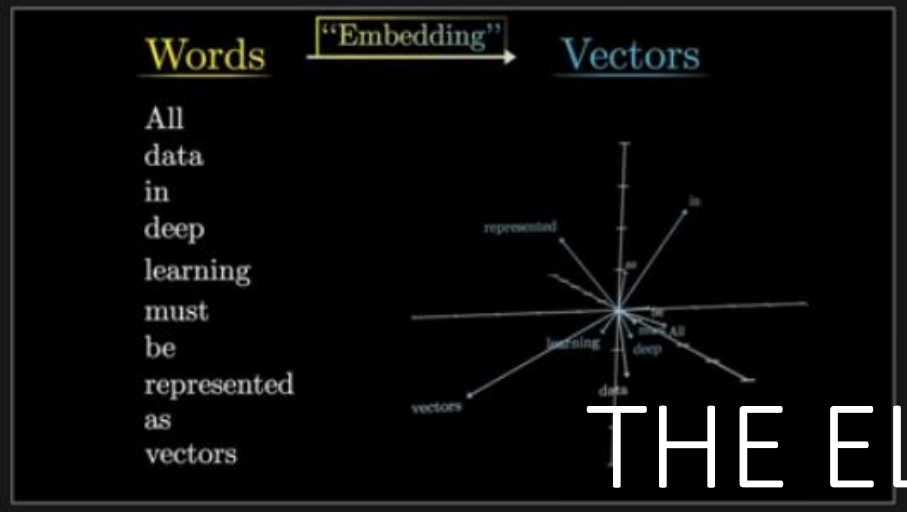


Transformer

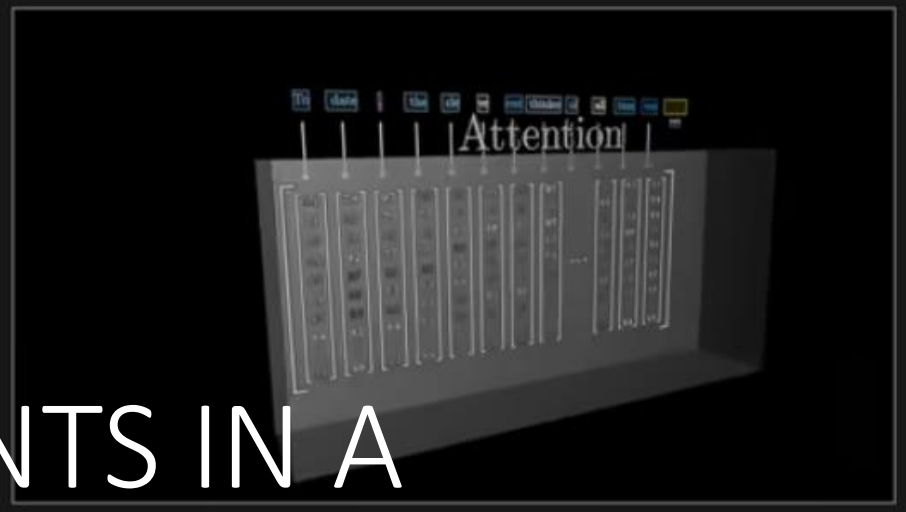
land	22%
forest	9%
country	5%
habitat	4%
forests	4%
soil	4%
territory	2%
woods	2%
lands	1%
waters	1%
woodland	1%
grass	1%
⋮	

THE ELEMENTS IN A TRANSFORMER

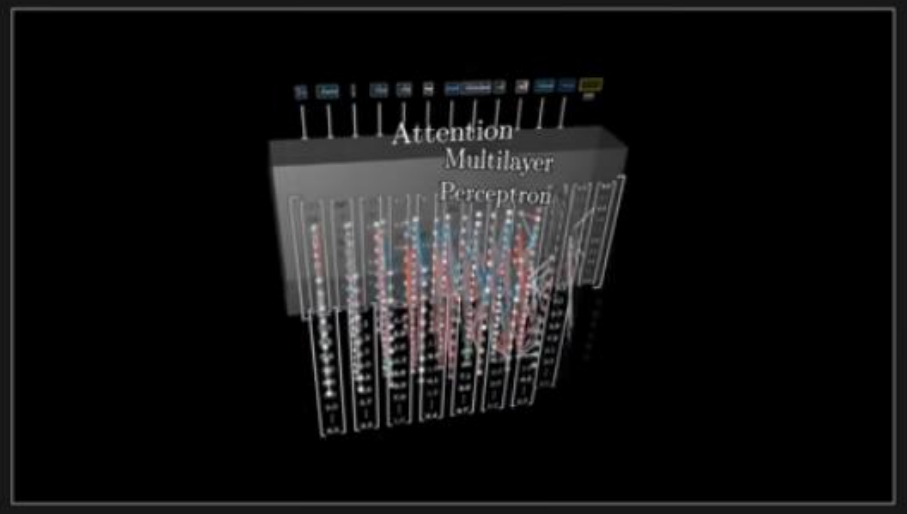
Embedding



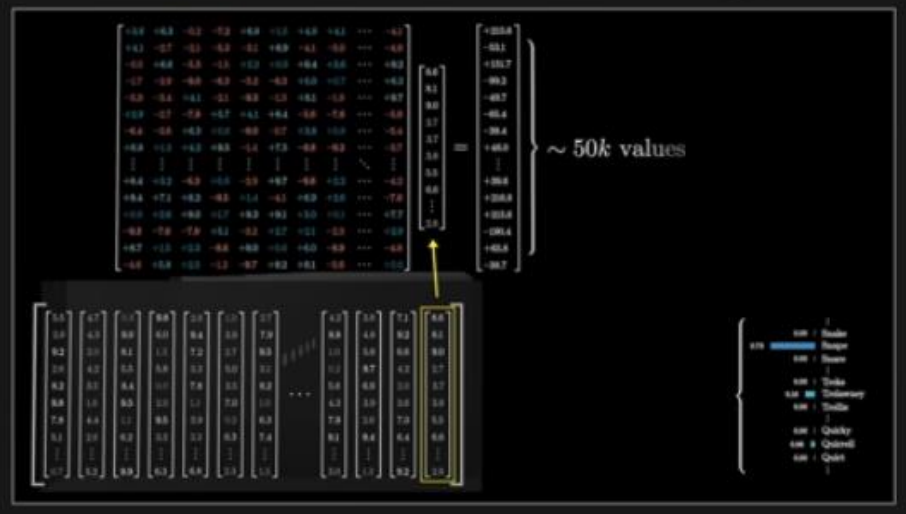
Attention



MLPs



Unembedding



Step 1: tokenize the text

To date, the cleverest thinker of all time was …

```
To | date | , | the | cle | ve | rest | thinker | of | all | time | was …
```

- Tokens are chunks of text (often subwords + punctuation).
- Tokenization is a pre-processing step, before the transformer blocks.
- The model predicts the next token, not necessarily the next “word.”

Step 2: map tokens to vectors

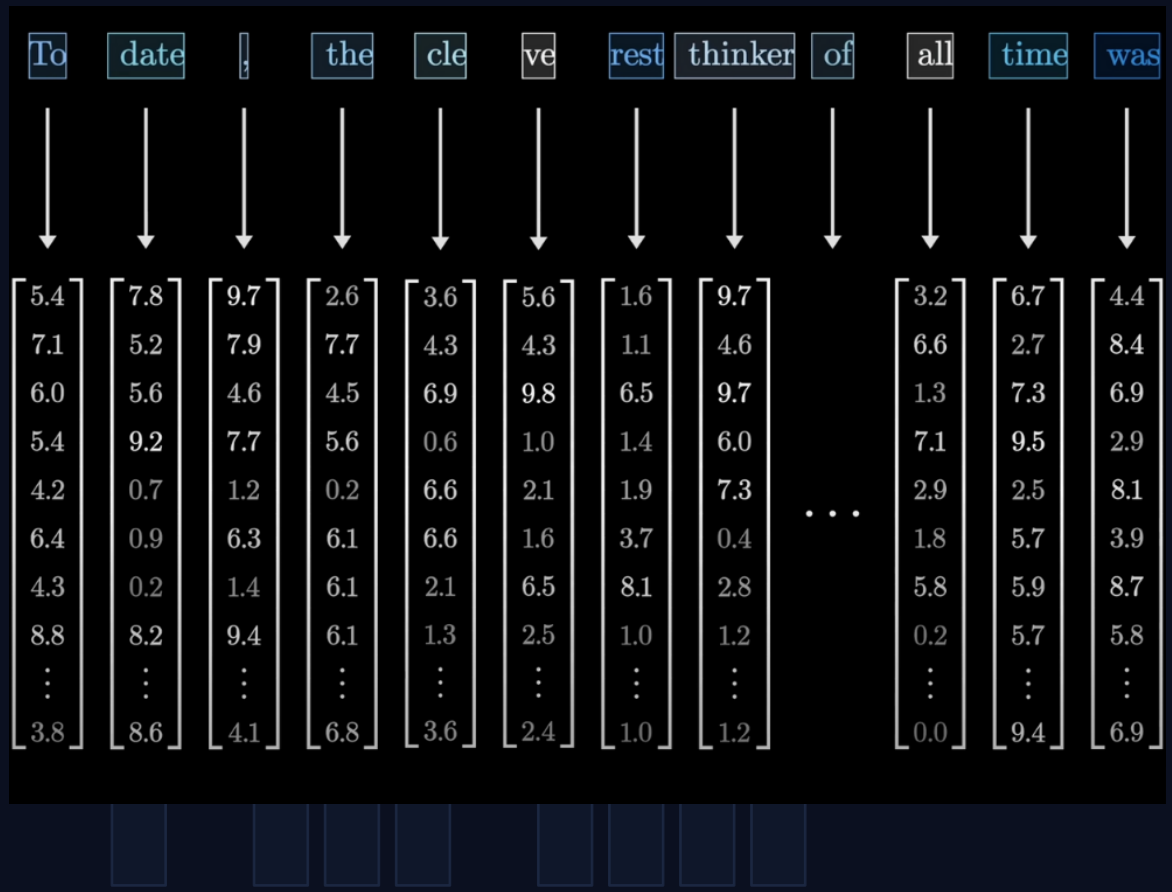
Embedding matrix as a learned lookup table

Vocabulary size V (≈ 50k)
Embedding dim d (≈ 12k in GPT-3)

Each token index selects a column vector. The entries start random and are tuned during training.

Parameter count (GPT-3 example)

```
V = 50,257 tokens
d = 12,288
W_E = V × d = 617,558,016 weights
```



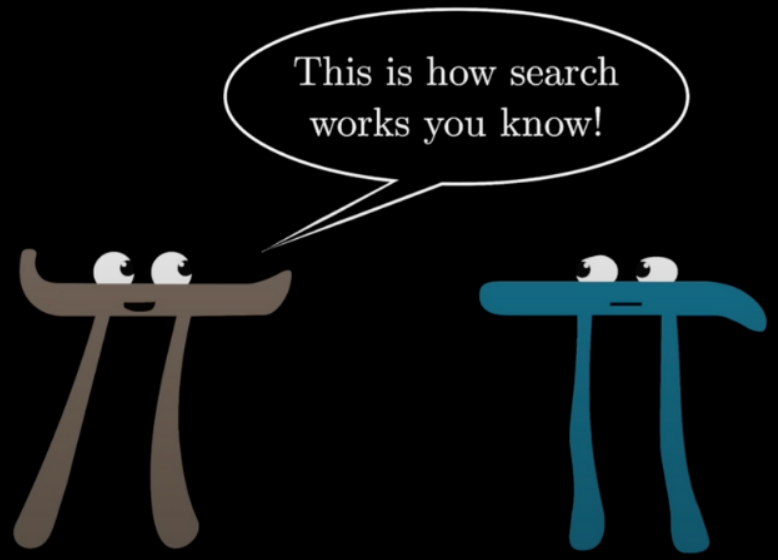
All words, ~ 50k

aah	aardvark	aardwolf	aargh	ab	aback	abacterial	abacus	abalone	abandon	…	zygoid	zygomatic	zygomorphic	zygosis	zygote	zygotic	zyme	zymogen	zymosis	zzz
−8.6	−1.5	−4.8	+6.9	−9.2	+9.1	−2.9	−2.8	−9.6	−6.2	⋯	−2.0	+8.5	−7.9	+8.8	+7.3	−0.9	−3.4	−5.3	+2.3	−9.2
−9.6	−1.4	−8.5	−4.9	−5.5	−4.9	−7.3	−9.7	−7.6	+2.3	⋯	+9.4	+9.7	−1.8	−6.7	+2.7	−0.2	+9.7	−8.6	+5.6	−4.2
−5.1	+3.2	−5.0	+3.3	+0.3	−1.5	+1.1	−4.2	+4.1	−1.7	⋯	−2.8	+6.5	+8.4	−9.0	−5.3	−3.0	+6.2	+9.6	+9.3	+8.0
−4.0	+9.7	−5.0	−7.8	+8.9	−5.3	+3.8	−8.7	+4.6	+7.6	⋯	−4.5	−2.4	−2.5	+4.9	−5.2	−6.5	−1.0	−3.9	+6.7	−5.2
+0.0	+8.8	+2.7	+7.3	+8.7	+5.0	+3.9	+9.3	+9.8	−0.9	⋯	−8.5	−4.1	−6.9	−1.6	−7.3	+2.1	−2.3	+7.8	+9.3	+0.9
−4.5	+1.8	+7.8	−1.9	+1.0	−4.5	−0.9	−1.9	−5.0	+0.1	⋯	−3.8	−2.5	+0.5	+5.0	−3.3	+8.4	+7.2	−8.9	−4.9	−1.1
−7.8	−3.0	+4.7	+3.6	+2.4	+4.2	−5.8	−3.1	+3.5	+7.5	⋯	+0.9	−4.3	−9.3	+4.2	−9.7	−2.5	+0.6	+8.3	−8.1	−1.9
−9.4	−3.1	+2.4	−4.4	−5.7	−7.6	+1.5	+3.9	+3.4	+8.9	⋯	−9.8	+2.9	+2.0	+1.8	+9.2	−9.6	+3.9	+6.2	+0.2	−3.3
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
+5.8	−8.0	−1.1	+0.4	+3.8	−8.1	−5.4	−1.8	+2.4	+7.7	⋯	+2.4	−7.3	+9.5	+7.4	+0.1	+8.4	+0.8	+8.4	+6.5	+9.3

$W_E =$

Embedding matrix

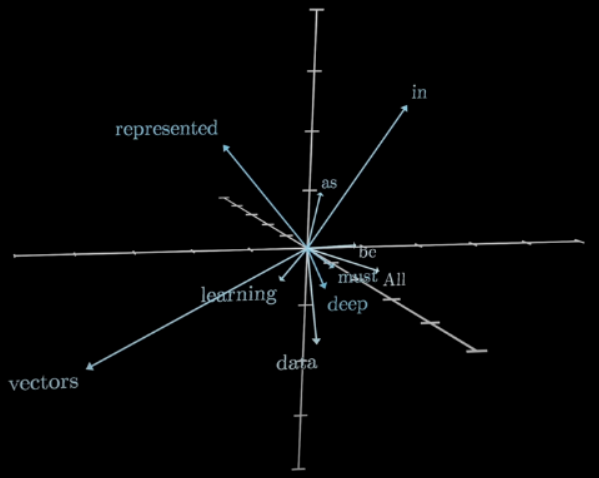
Embedding Matrix



Embedding Process

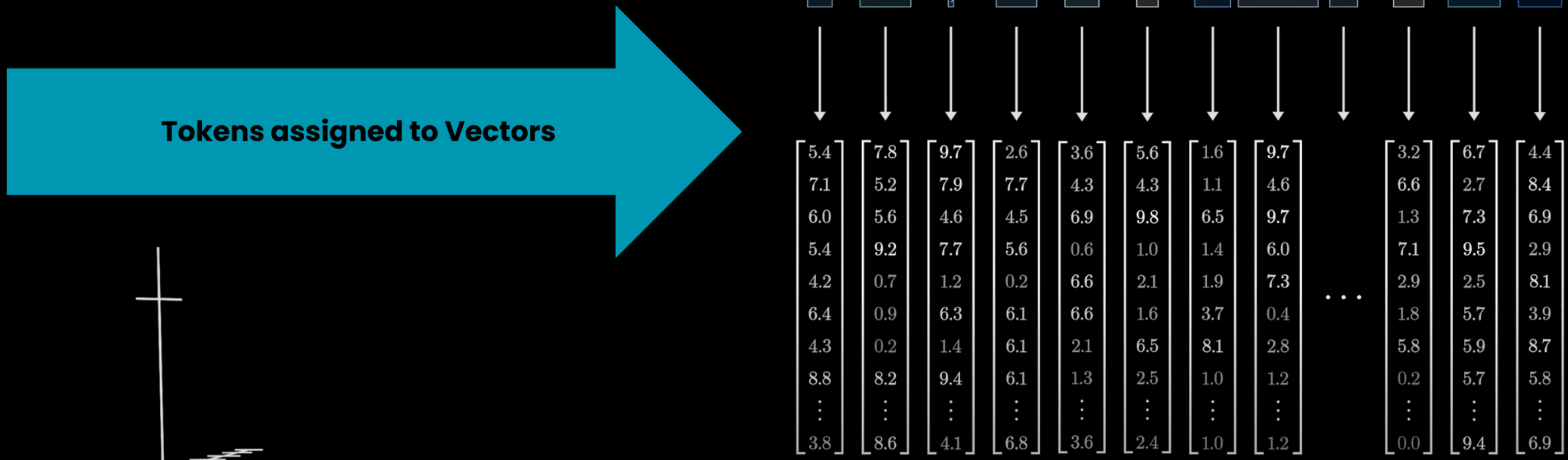
Words → "Embedding" → Vectors

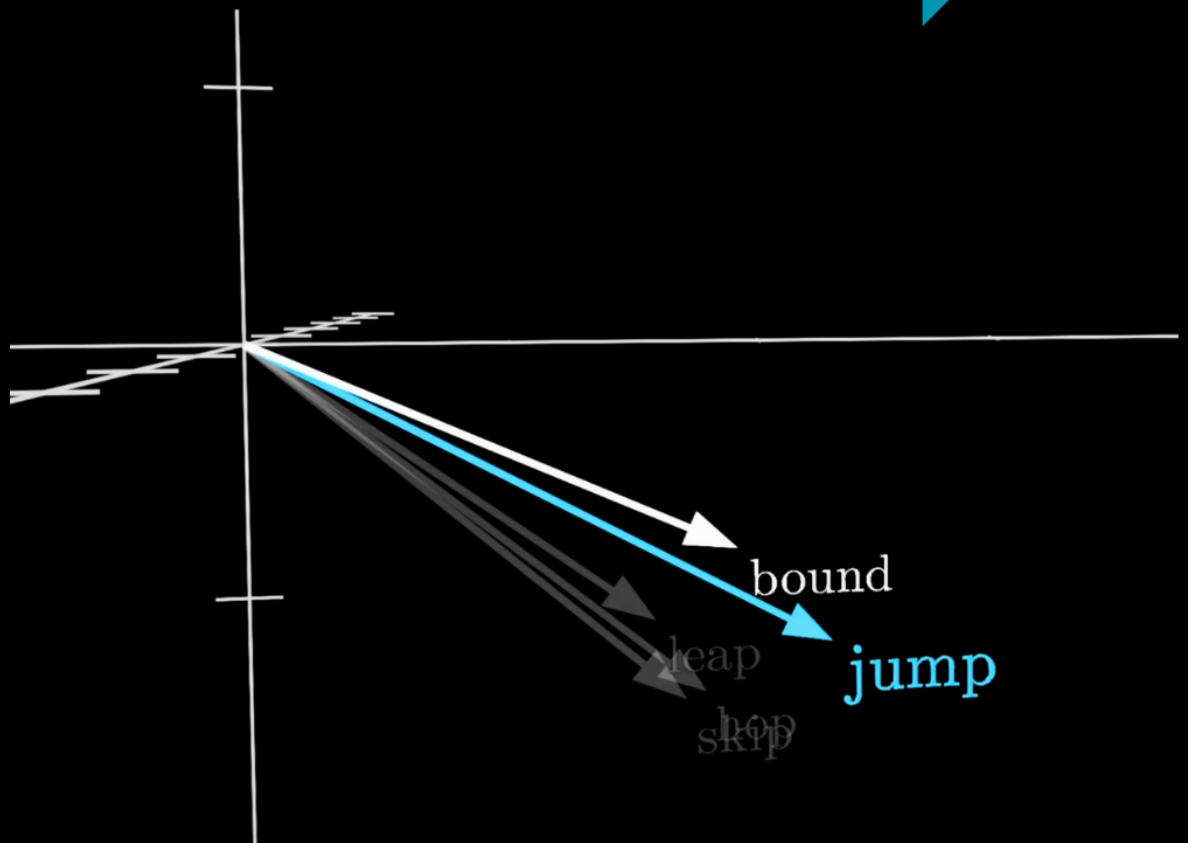
All
data
in
deep
learning
must
be
represented
as
vectors



Tokenization & Embedding

Tokens assigned to Vectors



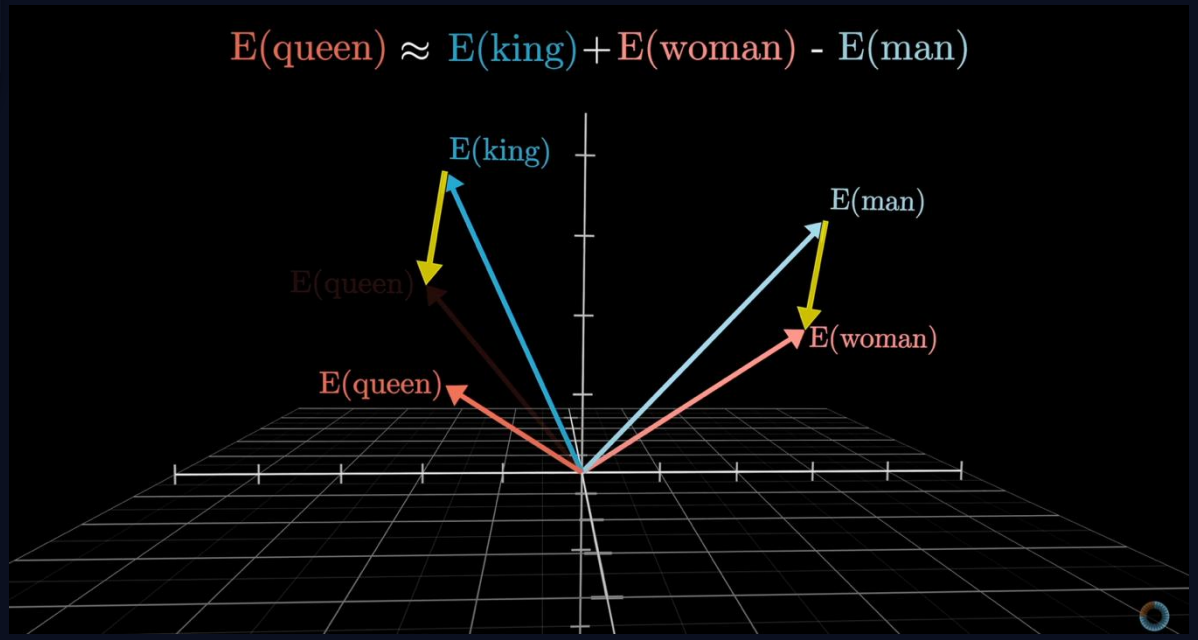


Words with similar meanings have close vectors

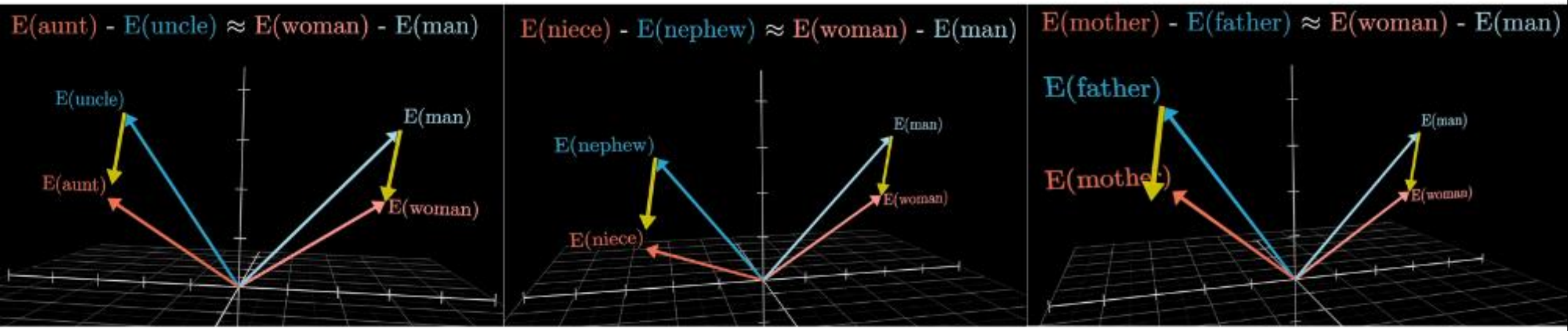
Geometric view

Think of each token vector as a point/direction in a high-dimensional space.

During training, models often arrange vectors so that *directions* correlate with attributes (gender, plurality, geography, etc.).



- Classic analogy structure (e.g., “king – man + woman ≈ queen”) is one way to *visualize* meaningful directions.
- (In reality, not every analogy works perfectly)


E(aunt) - E(uncle) ≈ E(woman) - E(man)
E(uncle)
E(man)
E(aunt)
E(woman)
E(niece) - E(nephew) ≈ E(woman) - E(man)
E(nephew)
E(man)
E(woman)
E(niece)
E(mother) - E(father) ≈ E(woman) - E(man)
E(father)
E(man)
E(woman)

Similarity via dot products

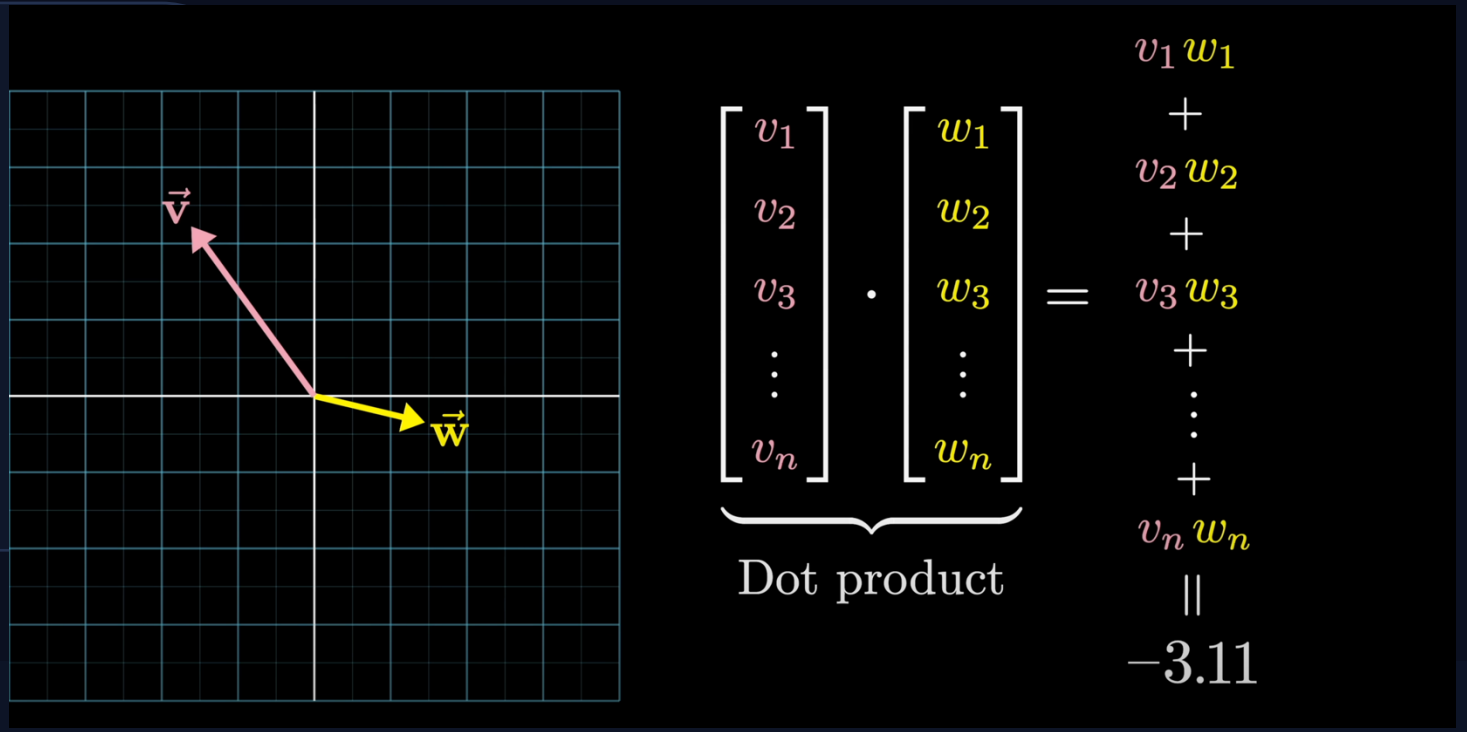
Dot product intuition

Geometrically: **v** · **w** is large when **v** and **w** point in similar directions, near 0 when orthogonal, negative when opposed.

Computationally: multiply component-wise and sum.

```
v · w = Σ_i v_i w_i
```

→ used as a similarity score inside attention



Step 3: transformer block (preview)

Each layer alternates two operations:

Attention: tokens “talk”

Vectors update based on *context*

Feed-forward (MLP): tokens “think”

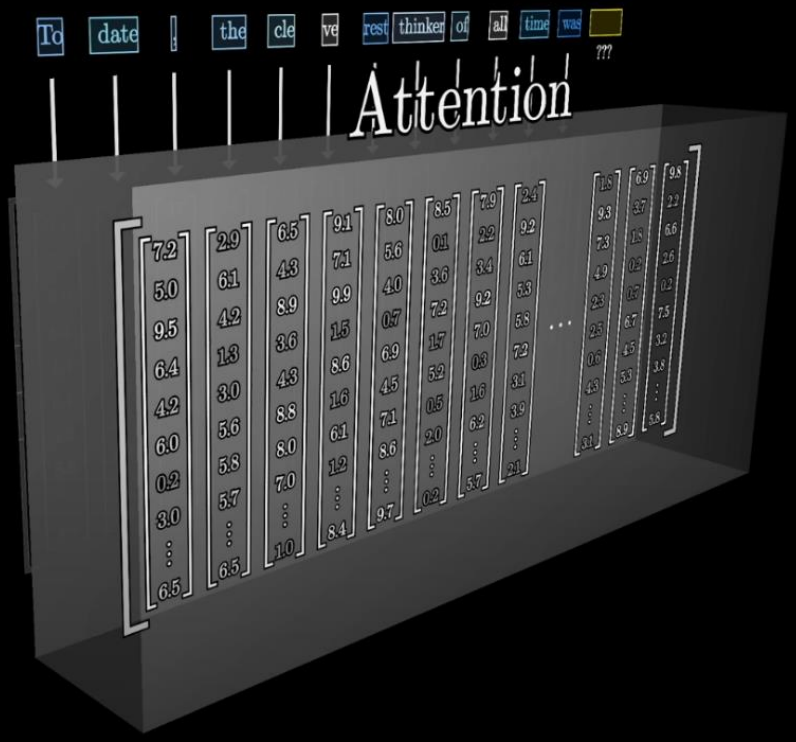
Same learned function applied to every position independently.

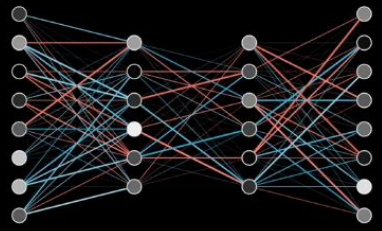
At the start, each token is just a lookup from the embedding matrix.
As many layers apply attention + MLP, the vector can be “tugged” into a nuanced direction representing meaning-in-context.

Deep = many layers

Attention Block

Attention Block





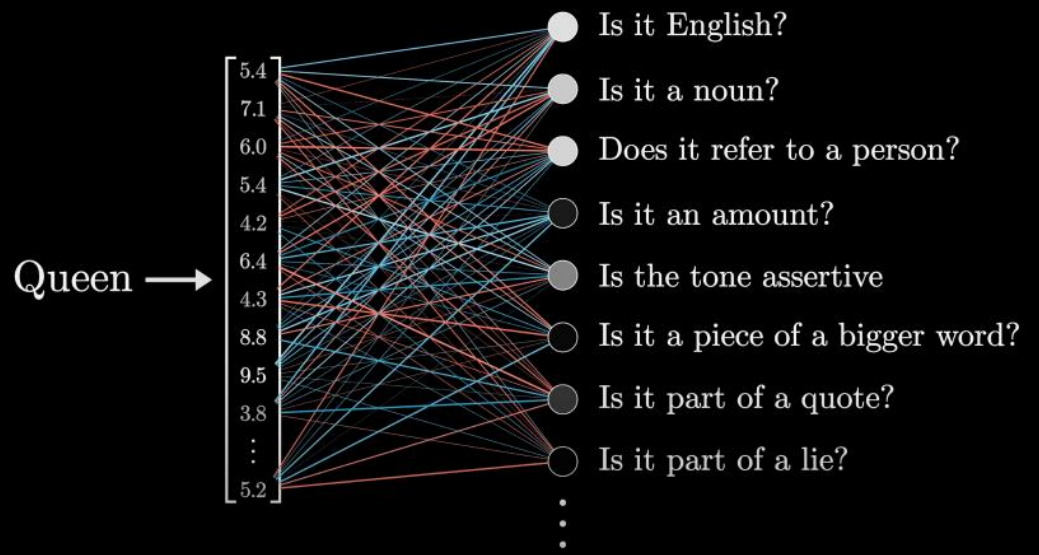
A machine learning model ...

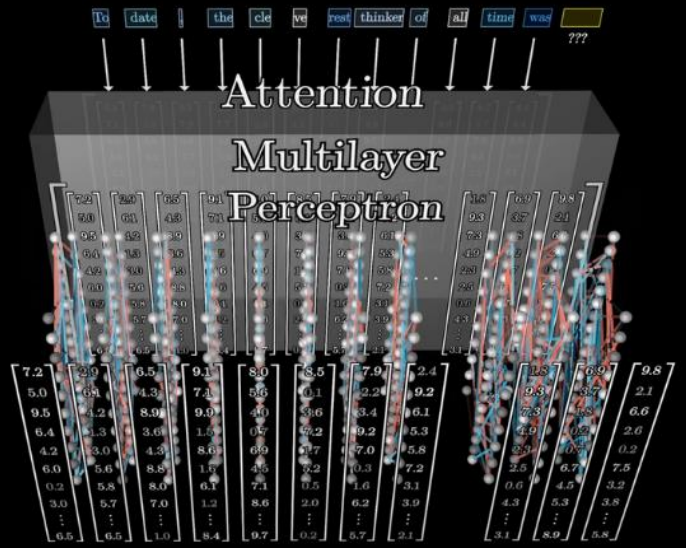
A fashion model ...

Example of Attention Block changing the vector of the word “model”

MLP Layer / Feedforward Layer

The Multilayer Perceptron asking "Questions" to Update the Vector





The Multilayer Perceptron "block"

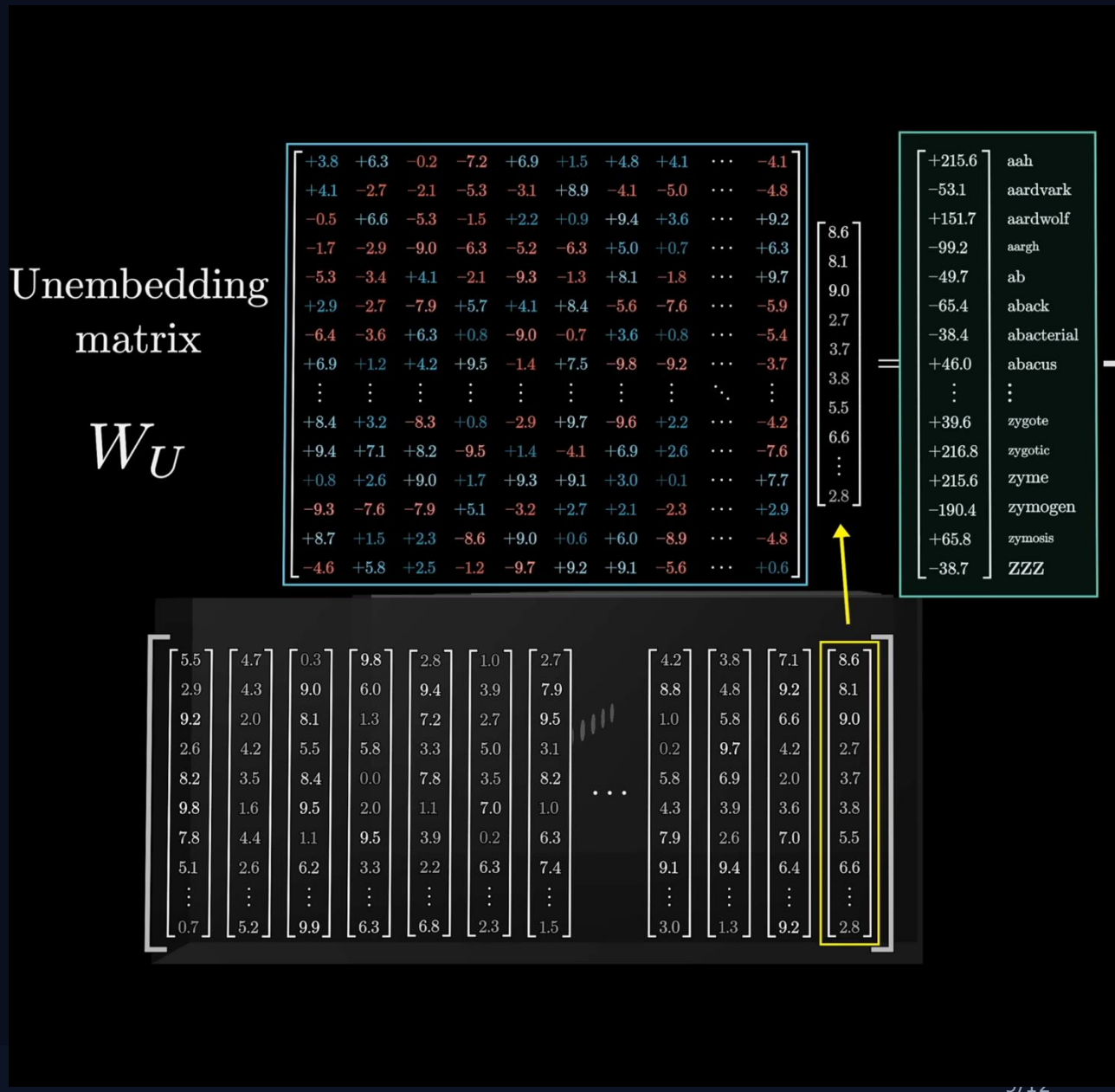
Step 4: Logits -- From vectors to next-token scores

Unembedding

At the end, map a context-rich vector back to vocabulary-sized scores.
These raw scores are called *logits*.

Why “the last vector”?

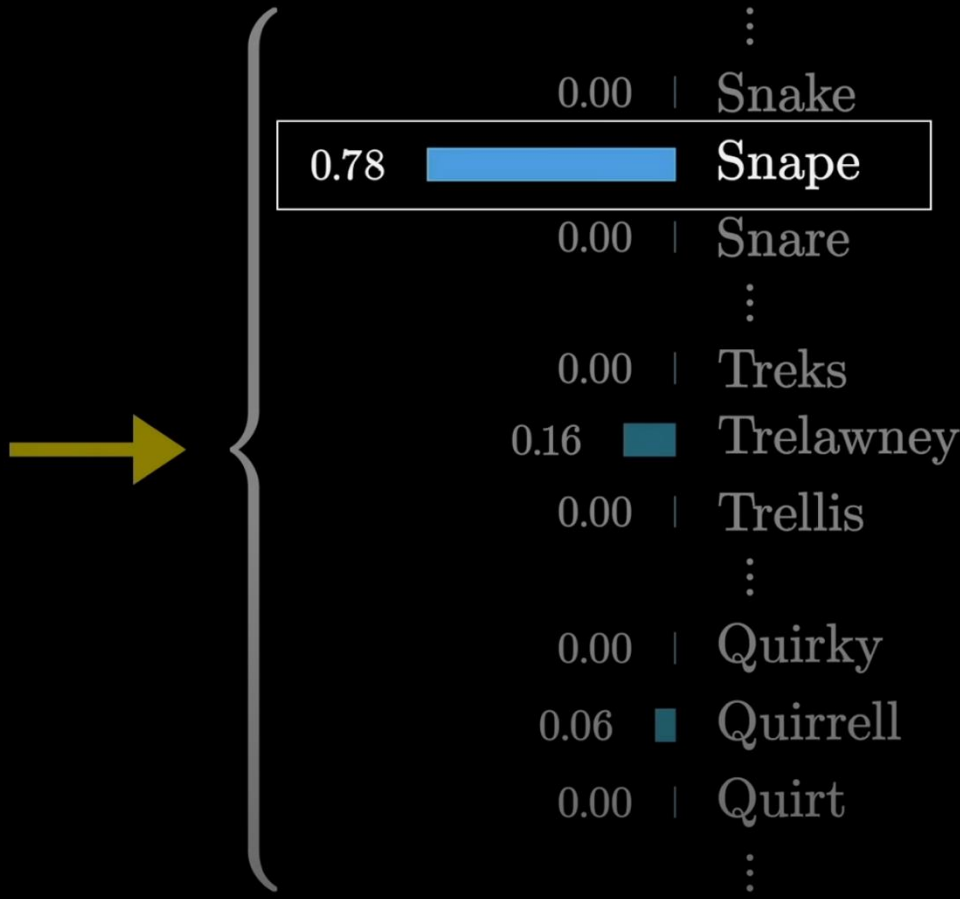
Causal language models are trained so every position predicts the token after it.
At generation time, we use the final position’s prediction as “next token.”



Step 5: softmax

Harry Potter was a highly unusual boy ... least favourite teacher, Professor [ ]

???

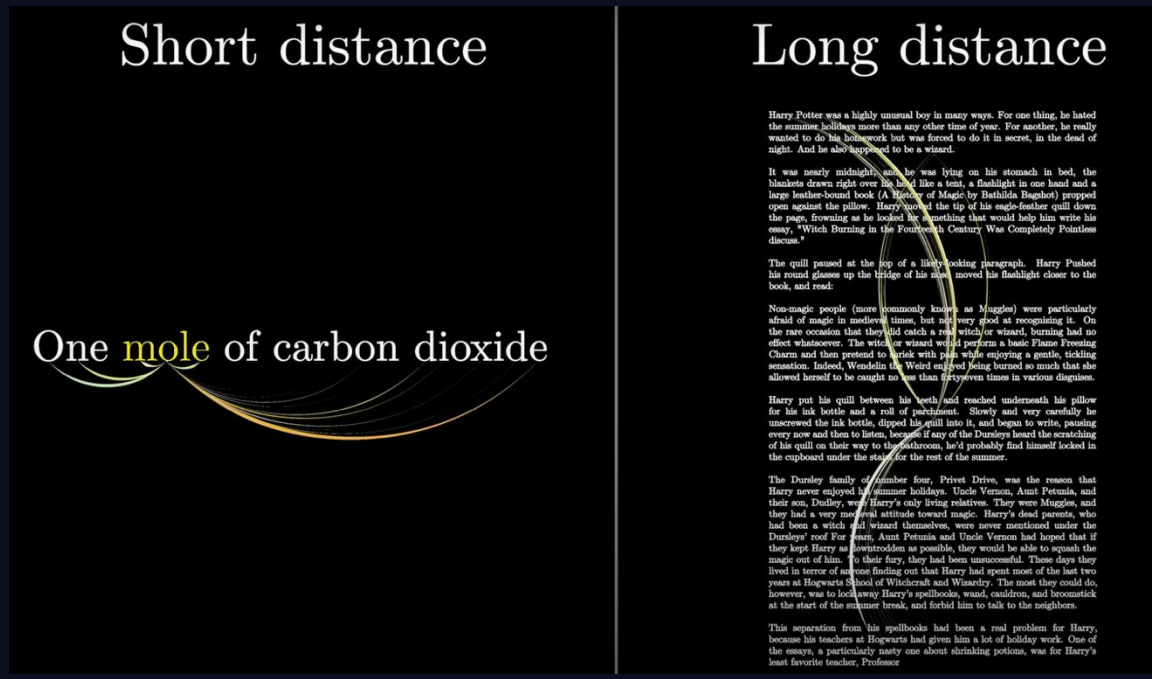


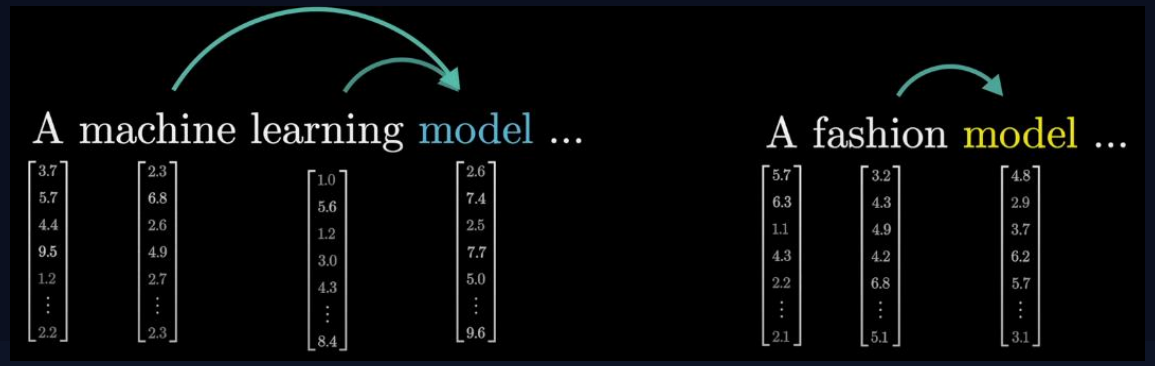
So far…

- Transformers predict next tokens; generation is repeated prediction + sampling.
- Tokens → embedding vectors → many layers of attention + MLP → logits → softmax.
- Embeddings live in a high-dimensional space where directions can carry meaning.
- Attention is the mechanism that updates vectors using context (next).

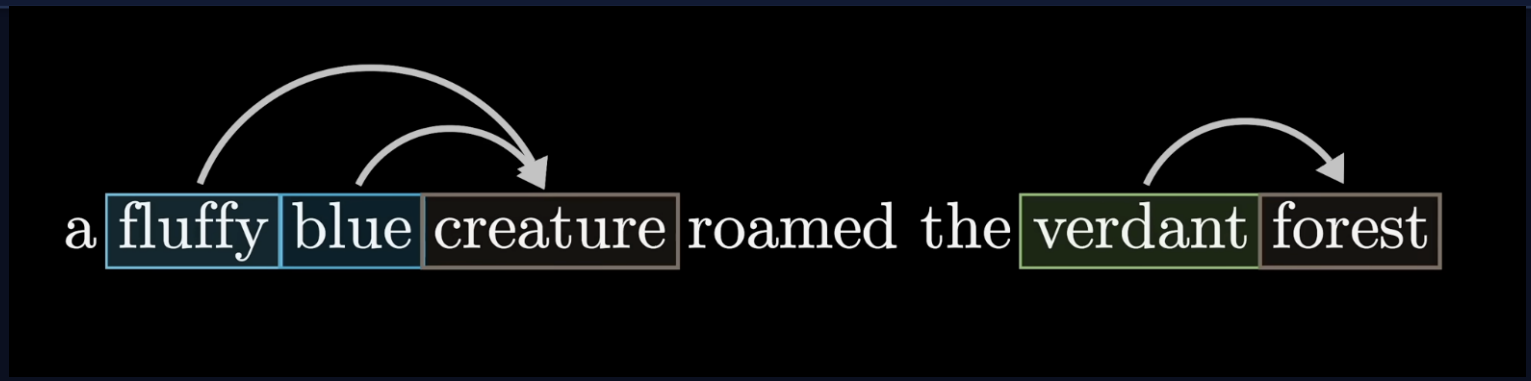
Desiderata

- route information between token embeddings over short and long distances
- handle information that's much richer than just a single word
- encode *all of the information* from the full context window that's relevant to predicting the next word.





Let’s start with an example sequence and toy example task:



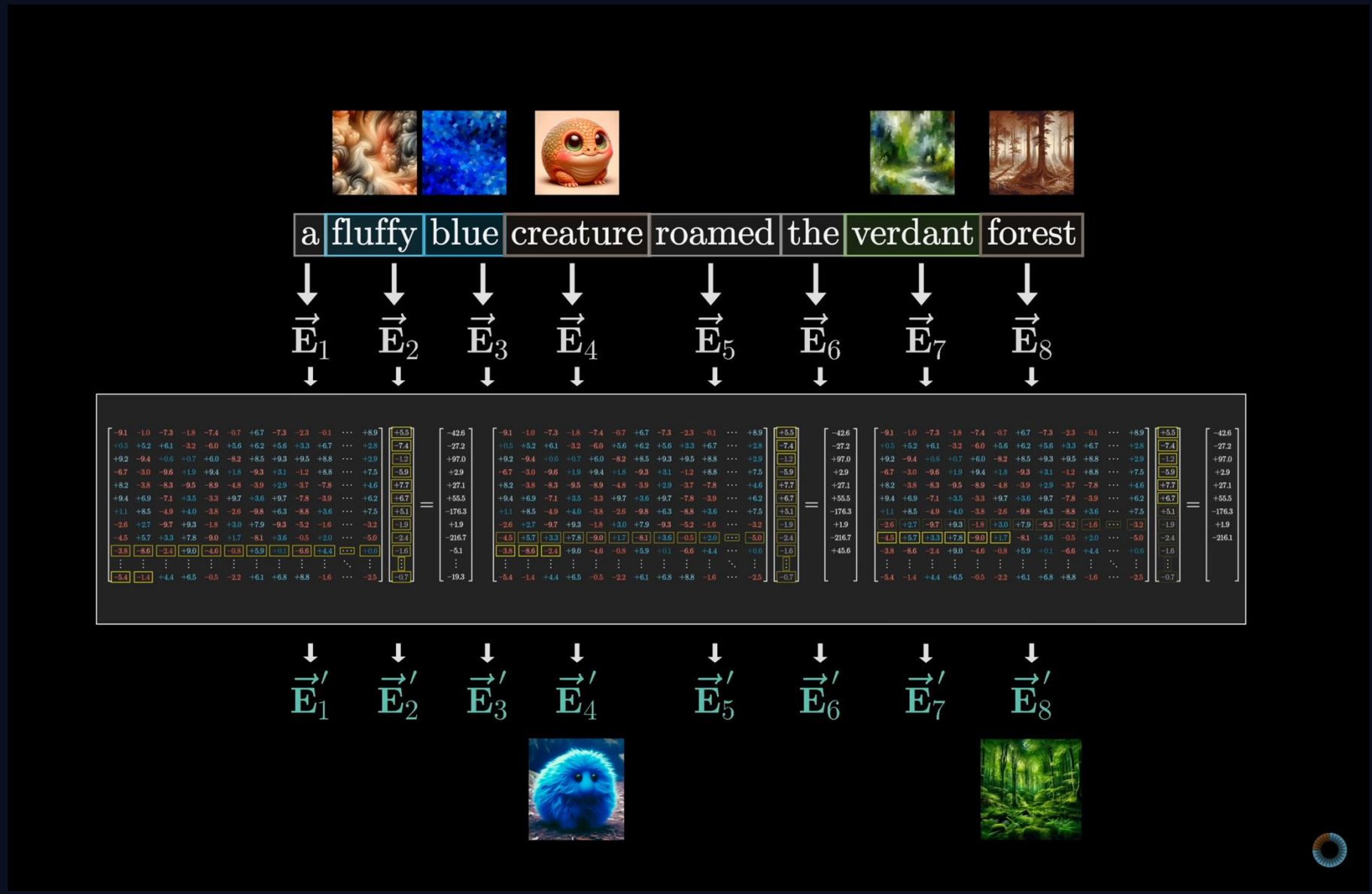
Toy Example Task: update the embeddings of this sequence so that the adjectives in the sequence above adjust the original embeddings of their corresponding nouns.

First, we tokenize then create the embeddings (which in practice also includes positional encoding).

Single-head attention

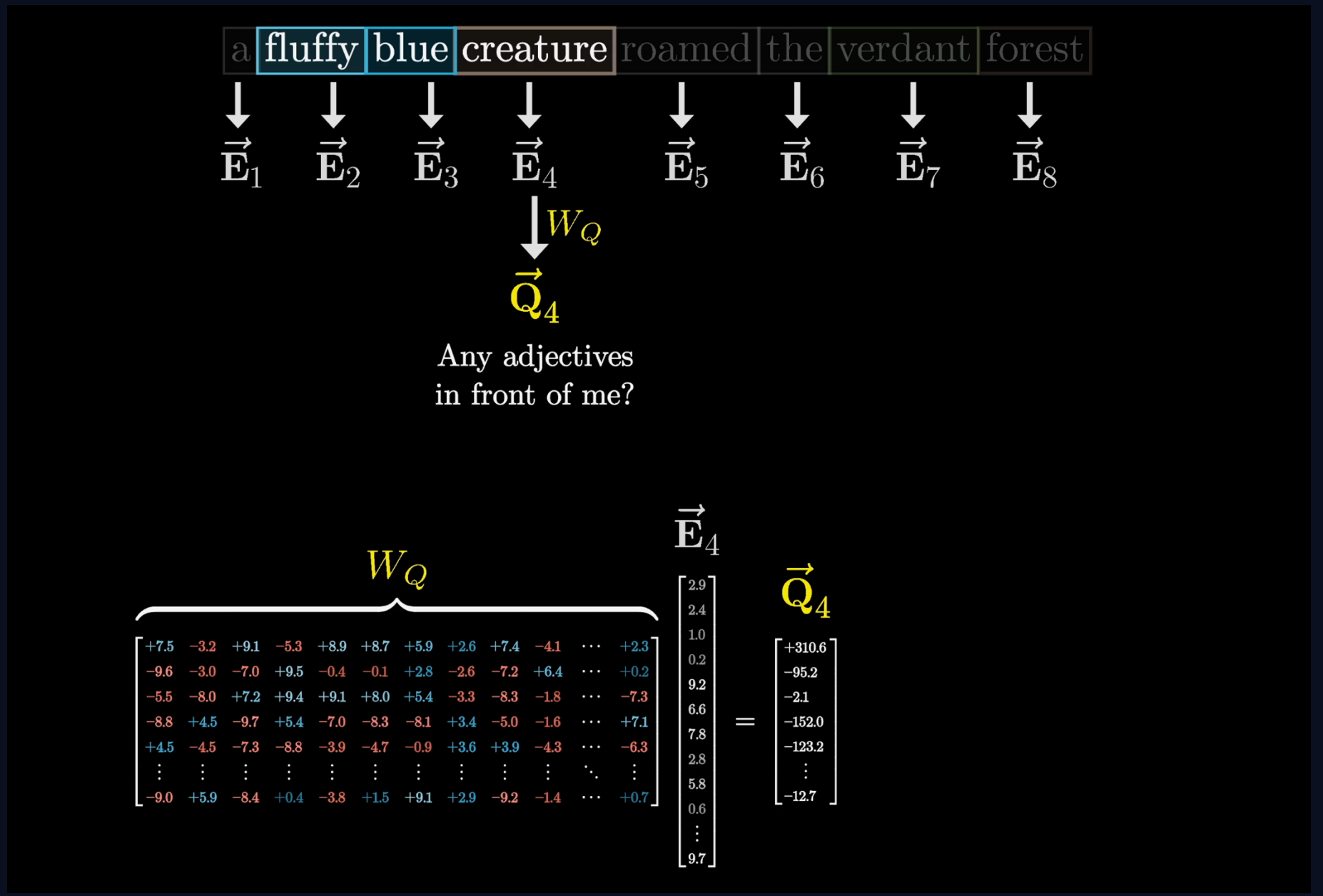
Goal of this “single-head attention” block:

Output E’ where the nouns have incorporated the meaning from their relevant adjectives



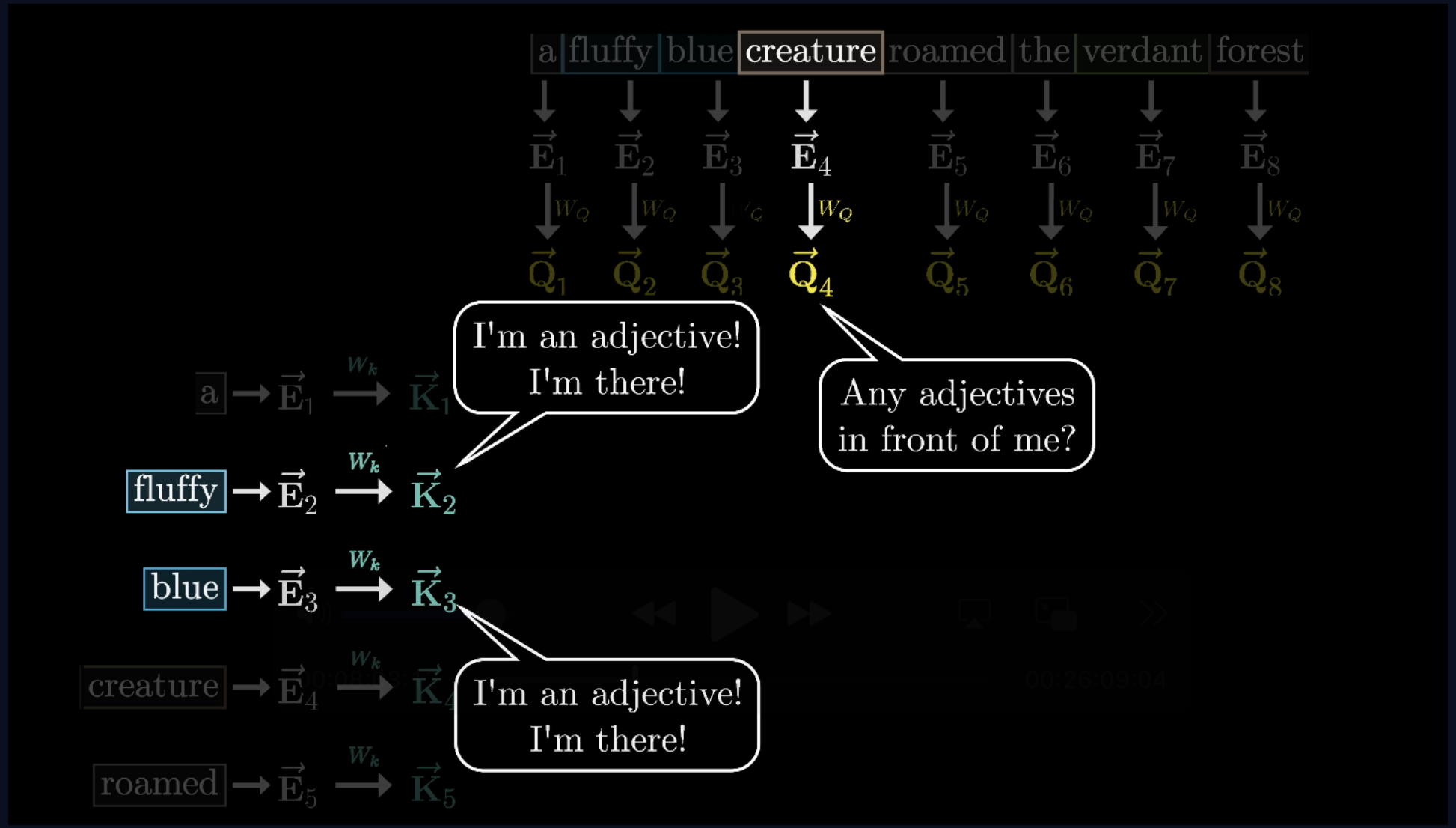
Query matrix:

maps the embeddings of nouns to a smaller query space that (somehow) encodes the notion of looking for adjectives in preceding positions.



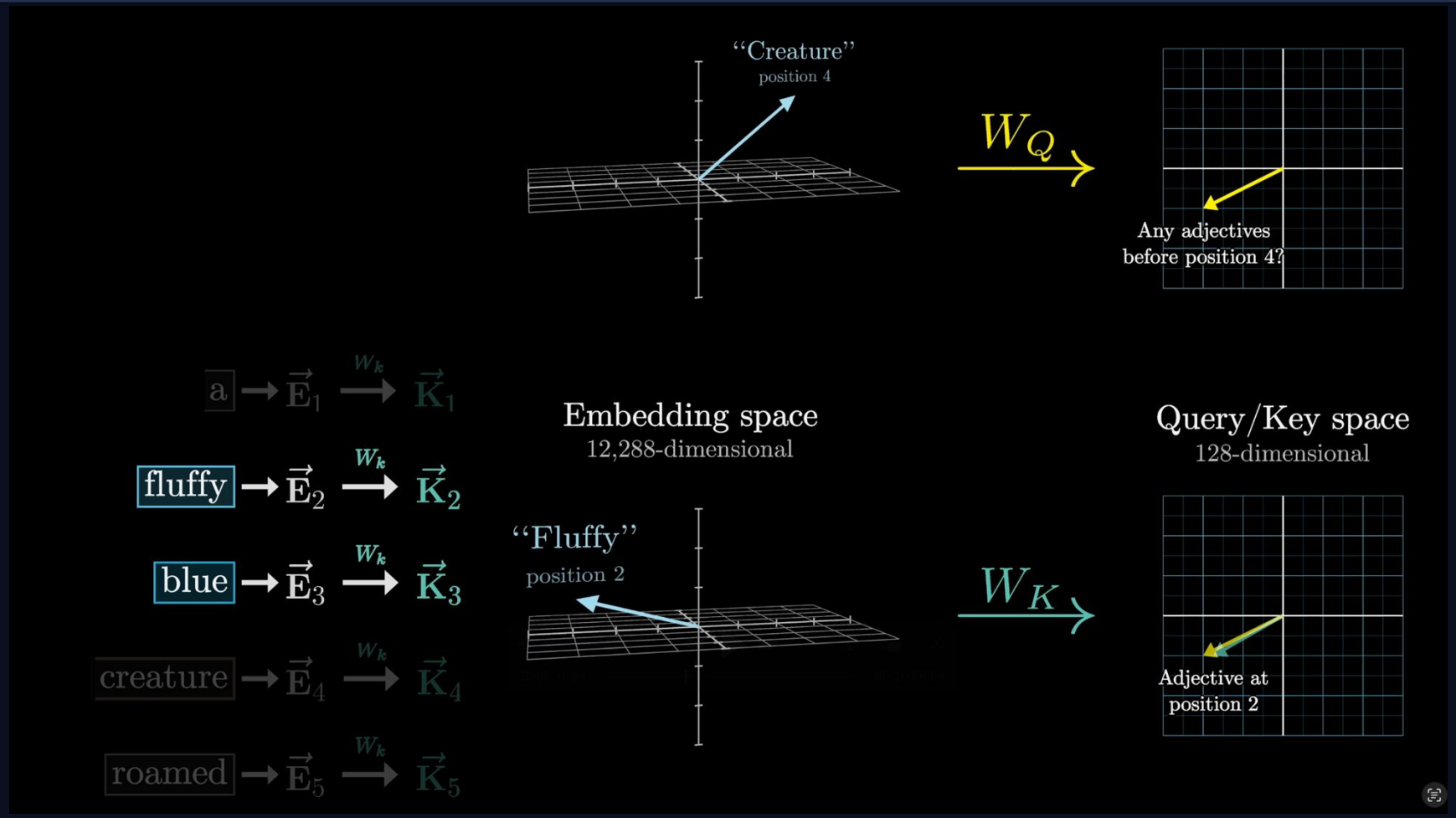
Key matrix:

maps the embeddings of nouns to a smaller key space that encodes the notion of how good of a match they are as to “answer” the query.



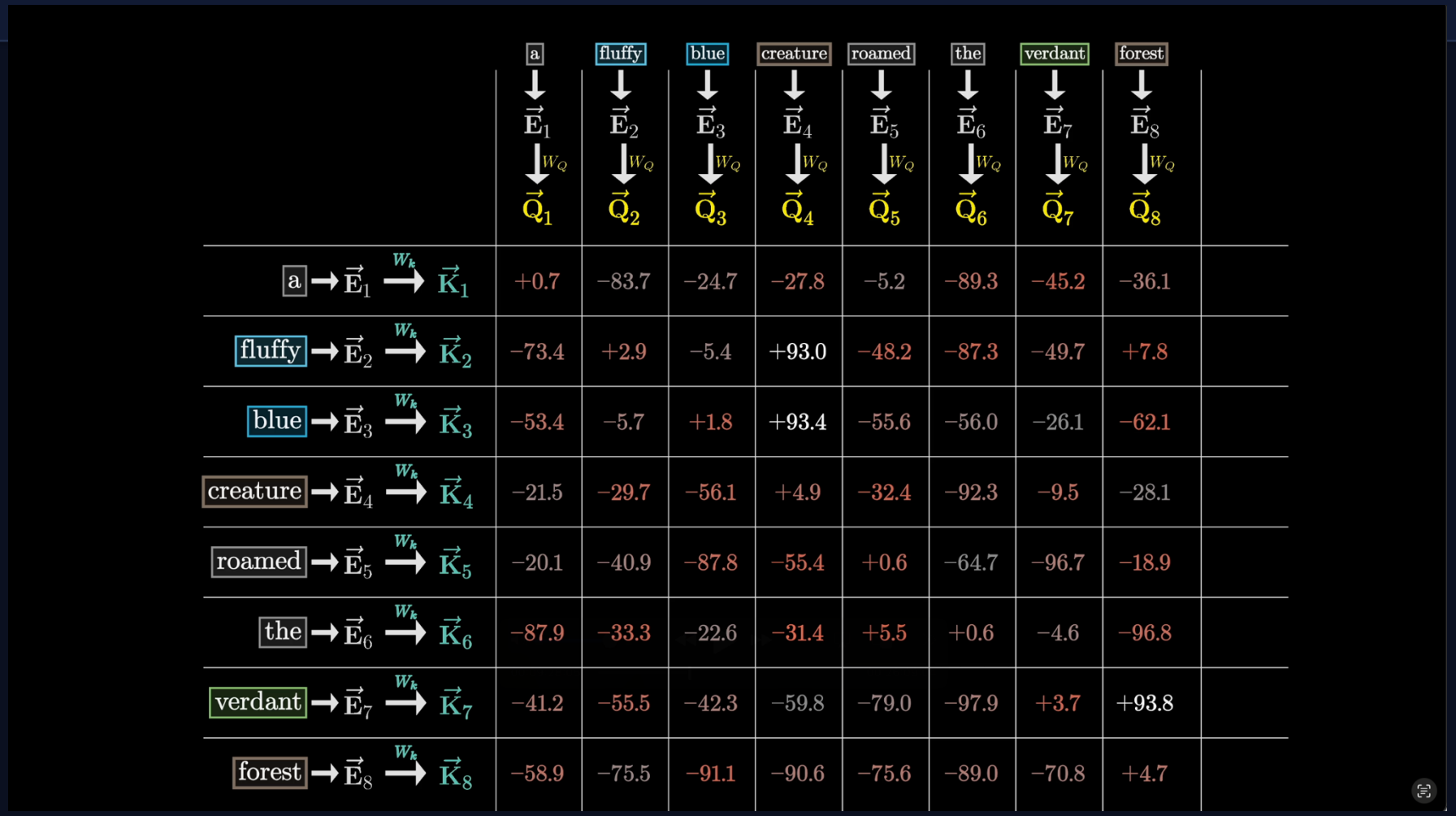
Dot-product self-attention

The way we measure how closely a given pair of key and query vectors align is to take their dot product. A larger dot product corresponds to stronger alignment.



Attention scores

Attention Scores = QK^T (pairwise similarity)
Row i: how token i attends to tokens 1…n

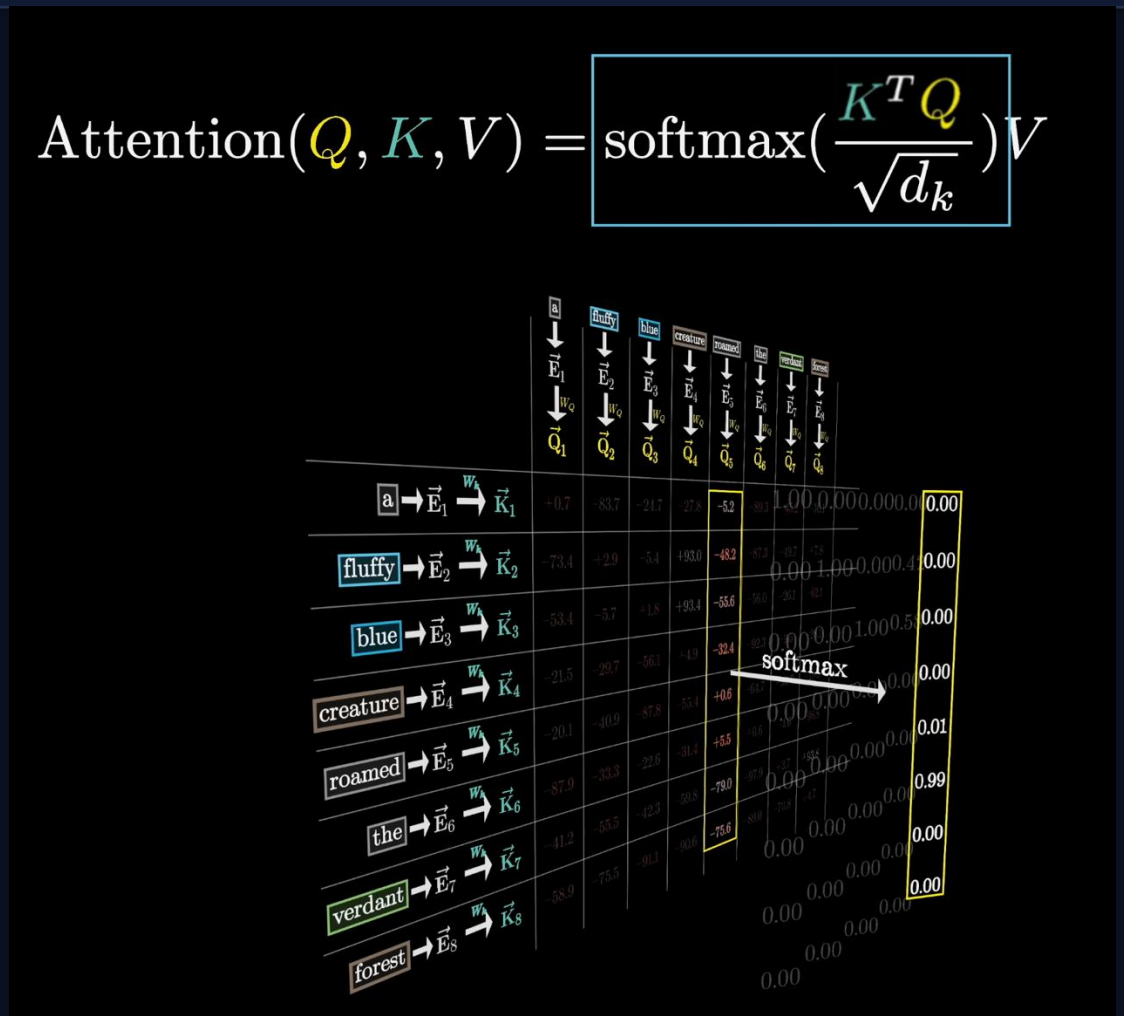


	a	fluffy	blue	creature	roamed	the	verdant	forest
	$\vec{E}_1 \xrightarrow{W_Q} \vec{Q}_1$	$\vec{E}_2 \xrightarrow{W_Q} \vec{Q}_2$	$\vec{E}_3 \xrightarrow{W_Q} \vec{Q}_3$	$\vec{E}_4 \xrightarrow{W_Q} \vec{Q}_4$	$\vec{E}_5 \xrightarrow{W_Q} \vec{Q}_5$	$\vec{E}_6 \xrightarrow{W_Q} \vec{Q}_6$	$\vec{E}_7 \xrightarrow{W_Q} \vec{Q}_7$	$\vec{E}_8 \xrightarrow{W_Q} \vec{Q}_8$
a $\to \vec{E}_1 \xrightarrow{W_k} \vec{K}_1$	+0.7	−83.7	−24.7	−27.8	−5.2	−89.3	−45.2	−36.1
fluffy $\to \vec{E}_2 \xrightarrow{W_k} \vec{K}_2$	−73.4	+2.9	−5.4	+93.0	−48.2	−87.3	−49.7	+7.8
blue $\to \vec{E}_3 \xrightarrow{W_k} \vec{K}_3$	−53.4	−5.7	+1.8	+93.4	−55.6	−56.0	−26.1	−62.1
creature $\to \vec{E}_4 \xrightarrow{W_k} \vec{K}_4$	−21.5	−29.7	−56.1	+4.9	−32.4	−92.3	−9.5	−28.1
roamed $\to \vec{E}_5 \xrightarrow{W_k} \vec{K}_5$	−20.1	−40.9	−87.8	−55.4	+0.6	−64.7	−96.7	−18.9
the $\to \vec{E}_6 \xrightarrow{W_k} \vec{K}_6$	−87.9	−33.3	−22.6	−31.4	+5.5	+0.6	−4.6	−96.8
verdant $\to \vec{E}_7 \xrightarrow{W_k} \vec{K}_7$	−41.2	−55.5	−42.3	−59.8	−79.0	−97.9	+3.7	+93.8
forest $\to \vec{E}_8 \xrightarrow{W_k} \vec{K}_8$	−58.9	−75.5	−91.1	−90.6	−75.6	−89.0	−70.8	+4.7

Scaled self-attention

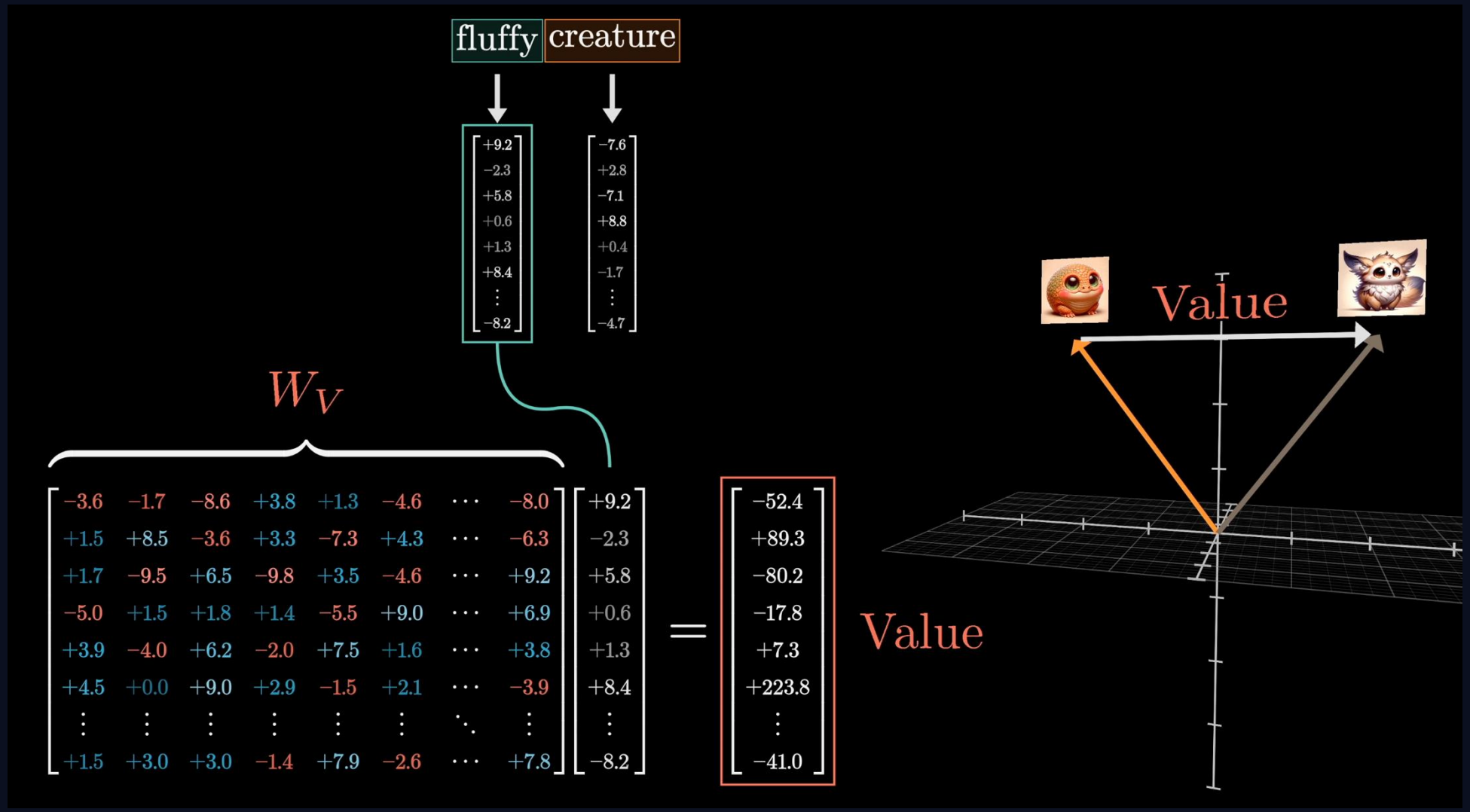
Scaled attention score for numerical stability: $QK^T)/\sqrt{d\_k}$
Softmax row-wise → attention weights summing to 1 like a probability distribution

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



Value matrix:

maps the embeddings to a higher-dimensional space that adjusts them to actually reflect their answer to the query's meaning.



Recap: What attention is doing

A GPT is trained to predict what comes next: a probability distribution over tokens.

Self-attention

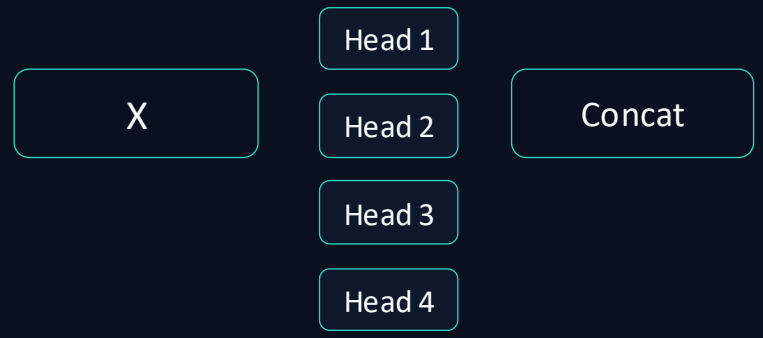
- Each token decides *who* to listen to
- It forms a weighted average of other tokens' information
- Weights depend on the input itself

Queries, Keys, Values (Q/K/V)

Query: what this token is looking for
Key: what this token advertises
Value: the information it contributes if attended to

Multi-head attention

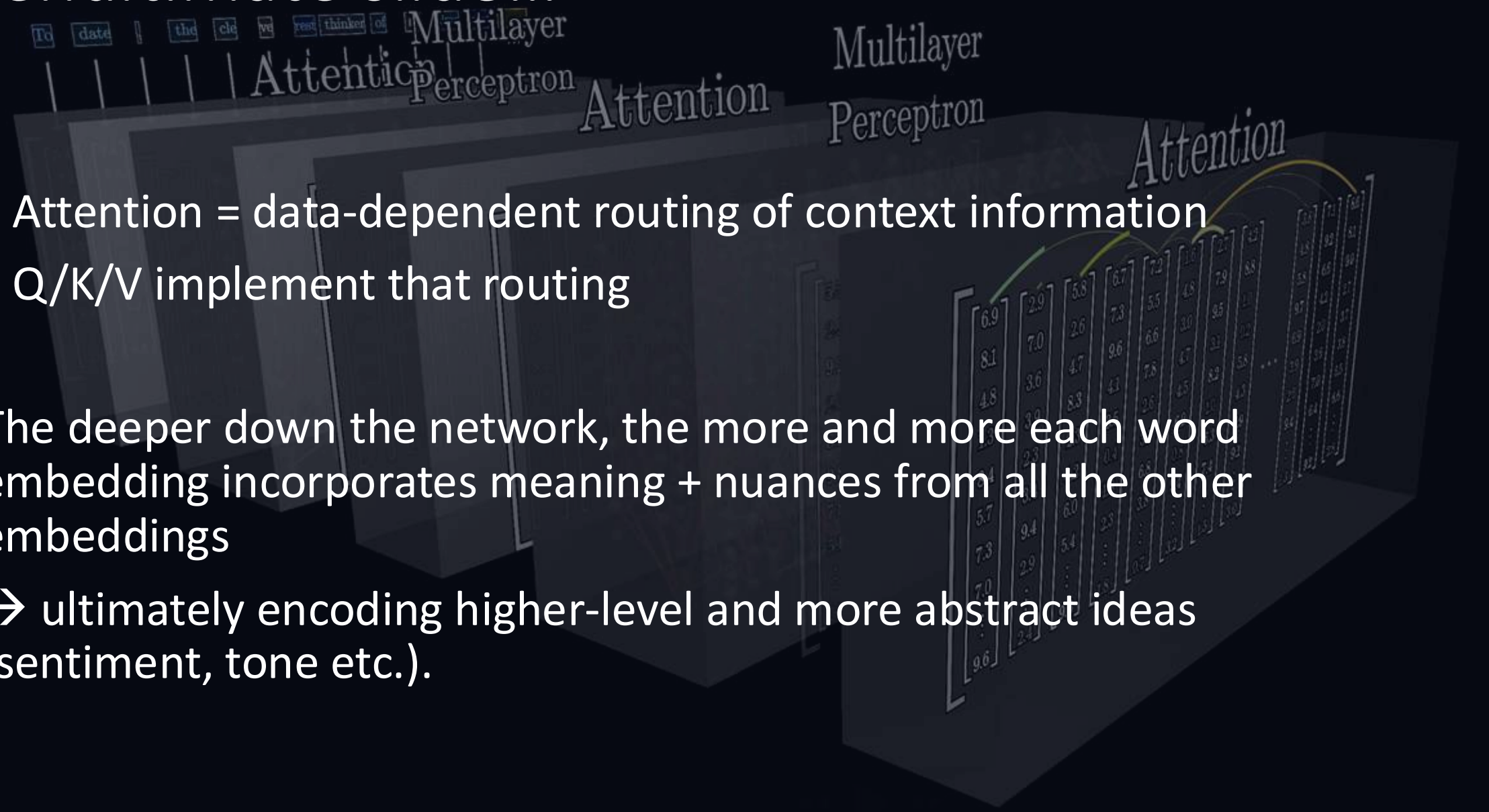
- Run several attentions in parallel (“heads”)
- Each head has its own W_Q, W_K, W_V
- Different heads learn different relations



Compute view:

- Attention is mostly batched matrix multiplications : QK^T then AV
- Parallel over tokens → GPU-friendly
- Cost grows ~$O(n^2)$ with sequence length

Penultimate slide…



- Attention = data-dependent routing of context information
- Q/K/V implement that routing

The deeper down the network, the more and more each word embedding incorporates meaning + nuances from all the other embeddings

→ ultimately encoding higher-level and more abstract ideas (sentiment, tone etc.).

“Attention is All You Need”

- Vaswani et al. 2017 (original Transformer paper)
- Brown et al. 2020 (GPT-3)