Natural Language Processing

LLMs, Transformer, PEFT, RAG, LLMs Project Life Cycle

KyawSwarTun

Most illustrations in these slides are from the internet, will be used for an educational purpose.

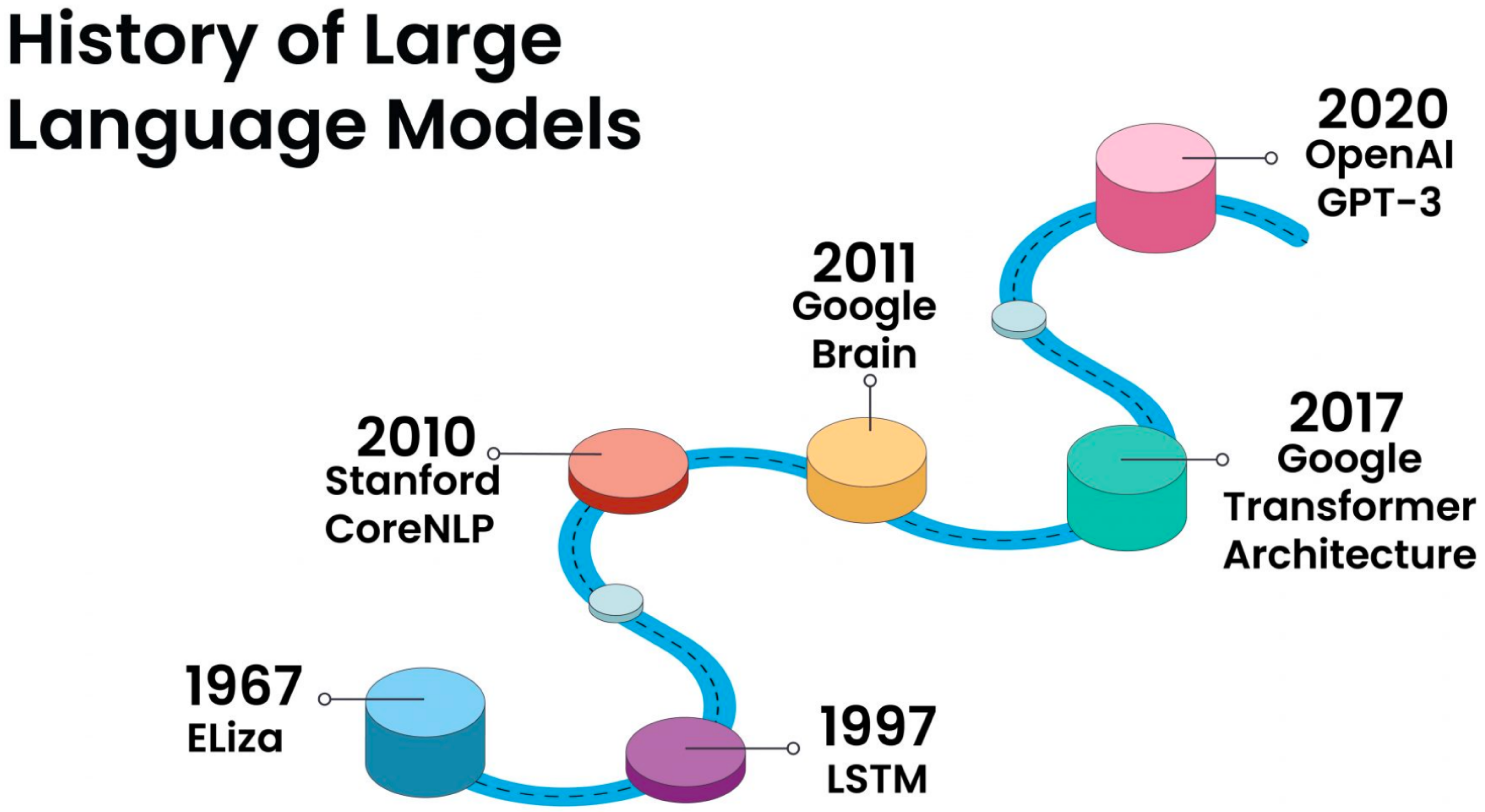


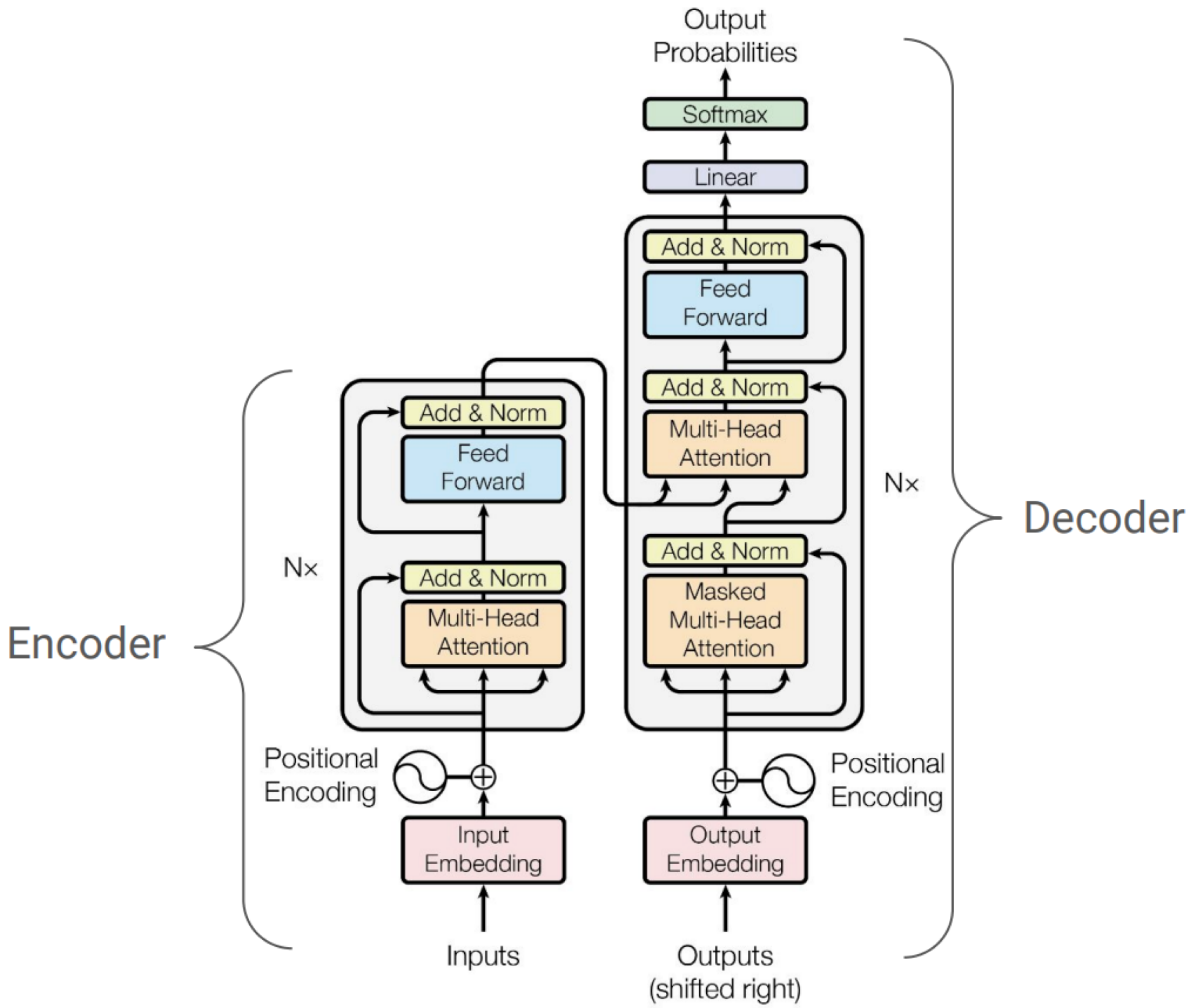
Fig. 1. History of Large Language Models from [2]

Transformer Block

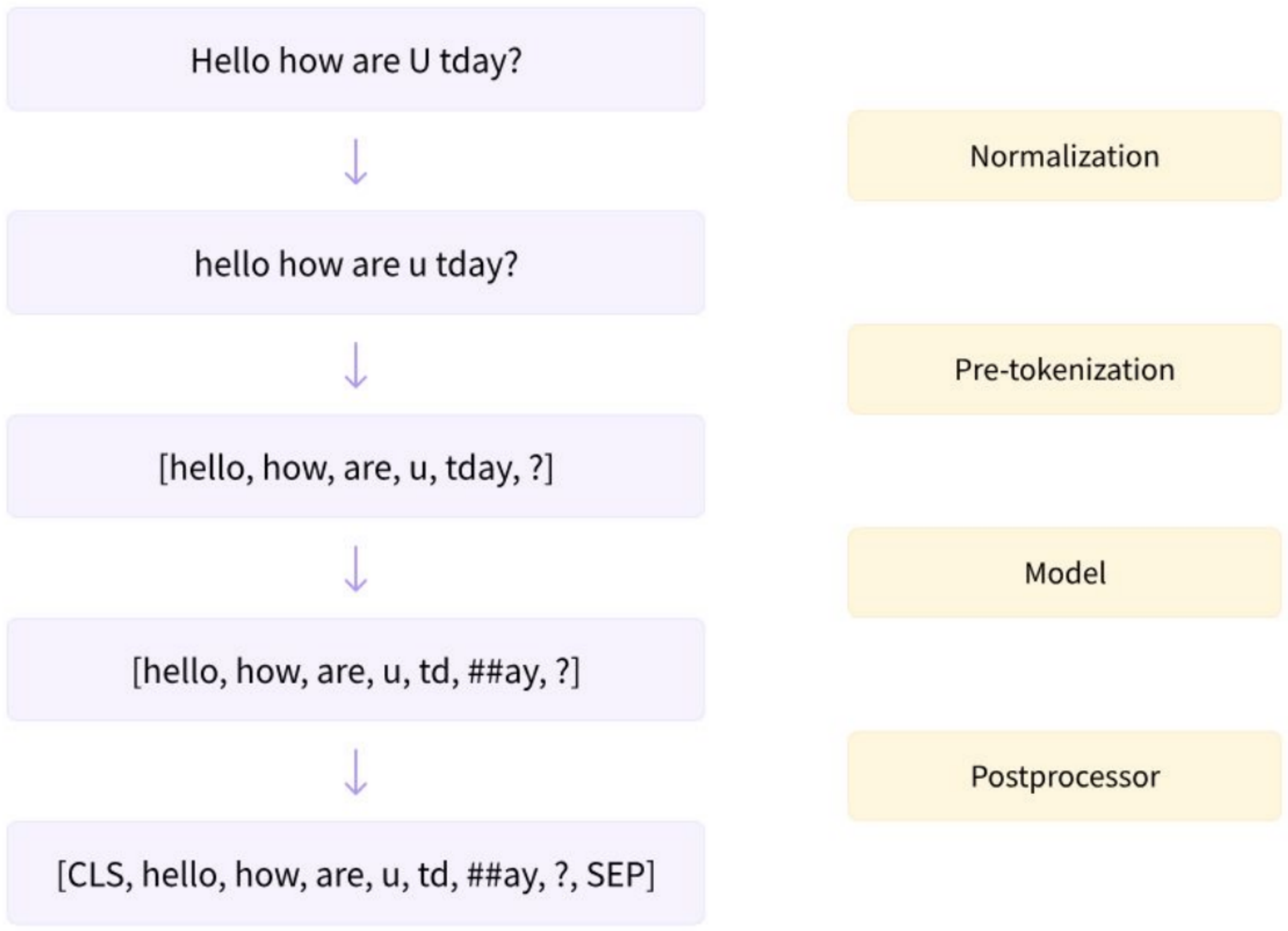
Next, we will learn exactly how the Transformer architecture works:

First, we will talk about the Encoder!

Next, we will go through the Decoder (which is quite similar)!



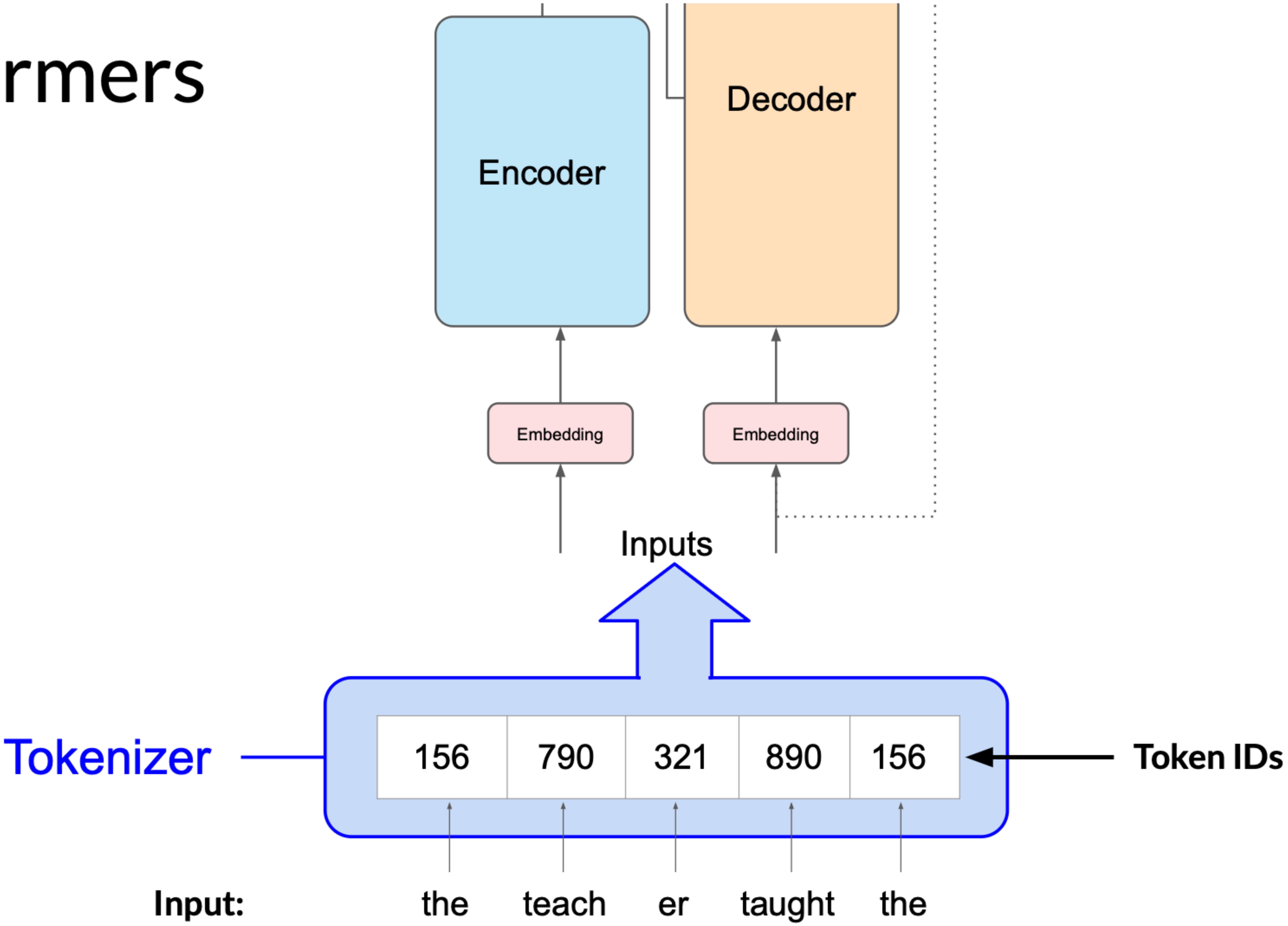
Tokenization: decomposing a sentence into a sequence of tokens



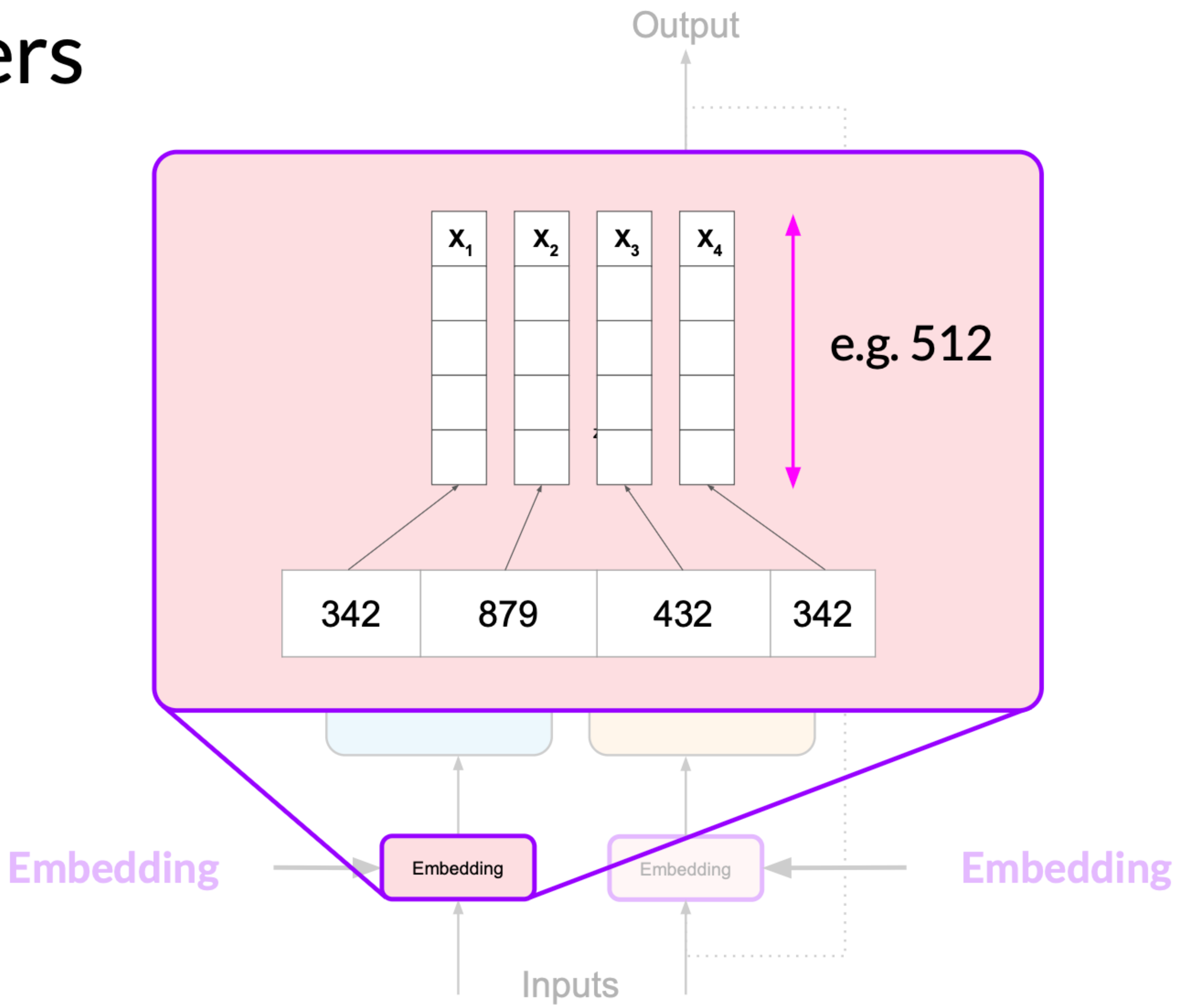
Every single explanation you will ever see about Language Models use **words**, **BUT** in reality the unit object is **tokens**

Words != Tokens

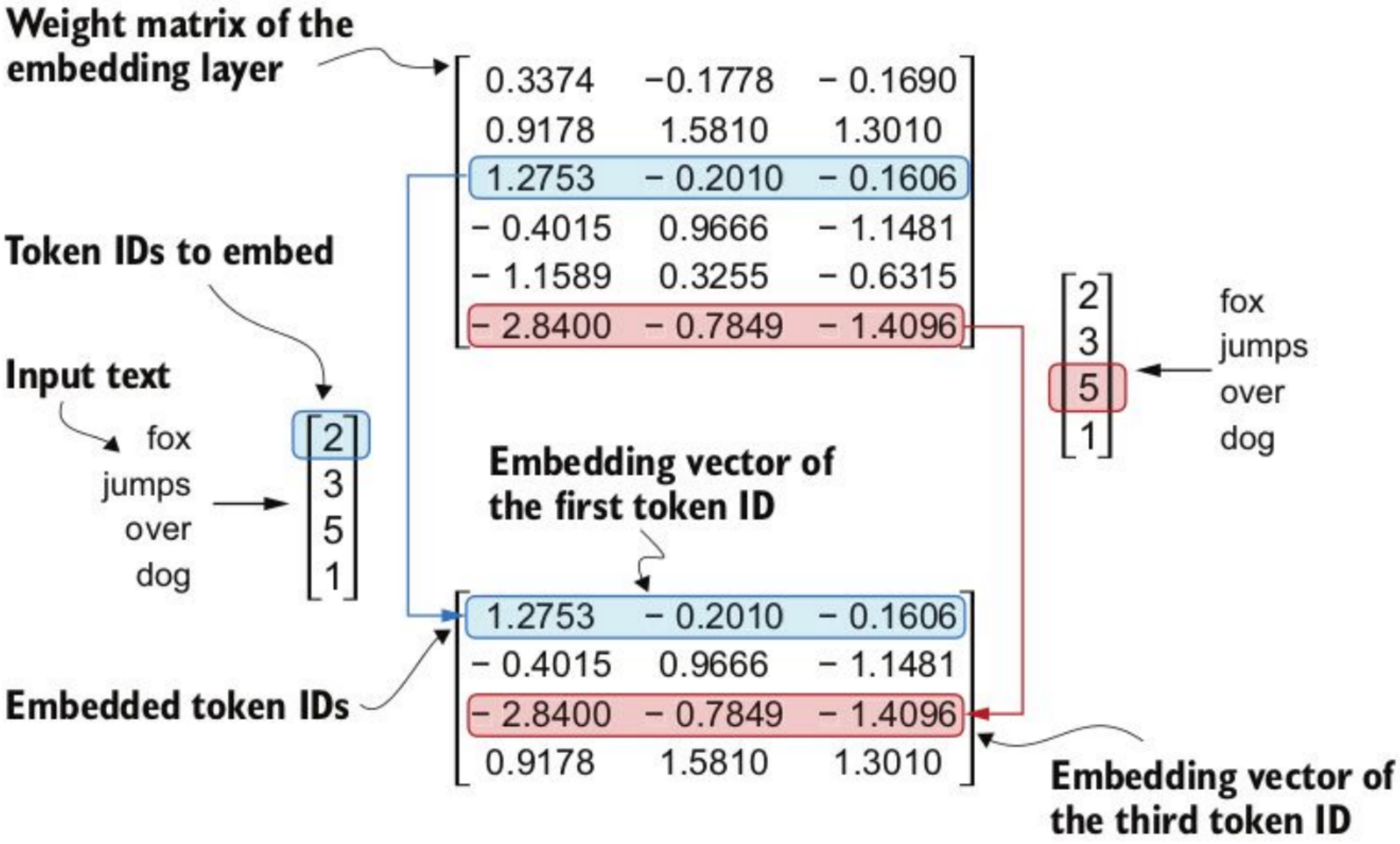
Transformers



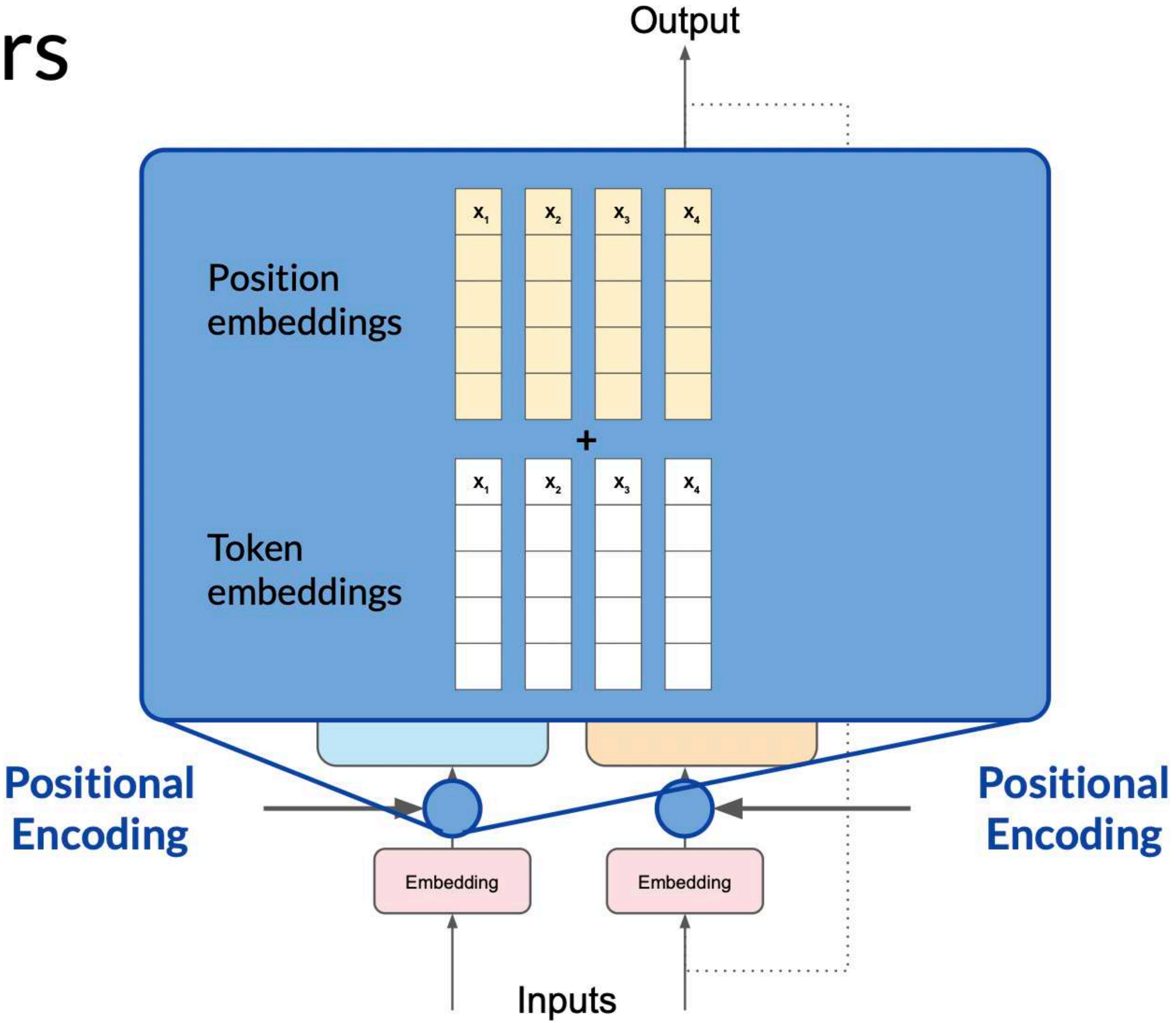
Transformers

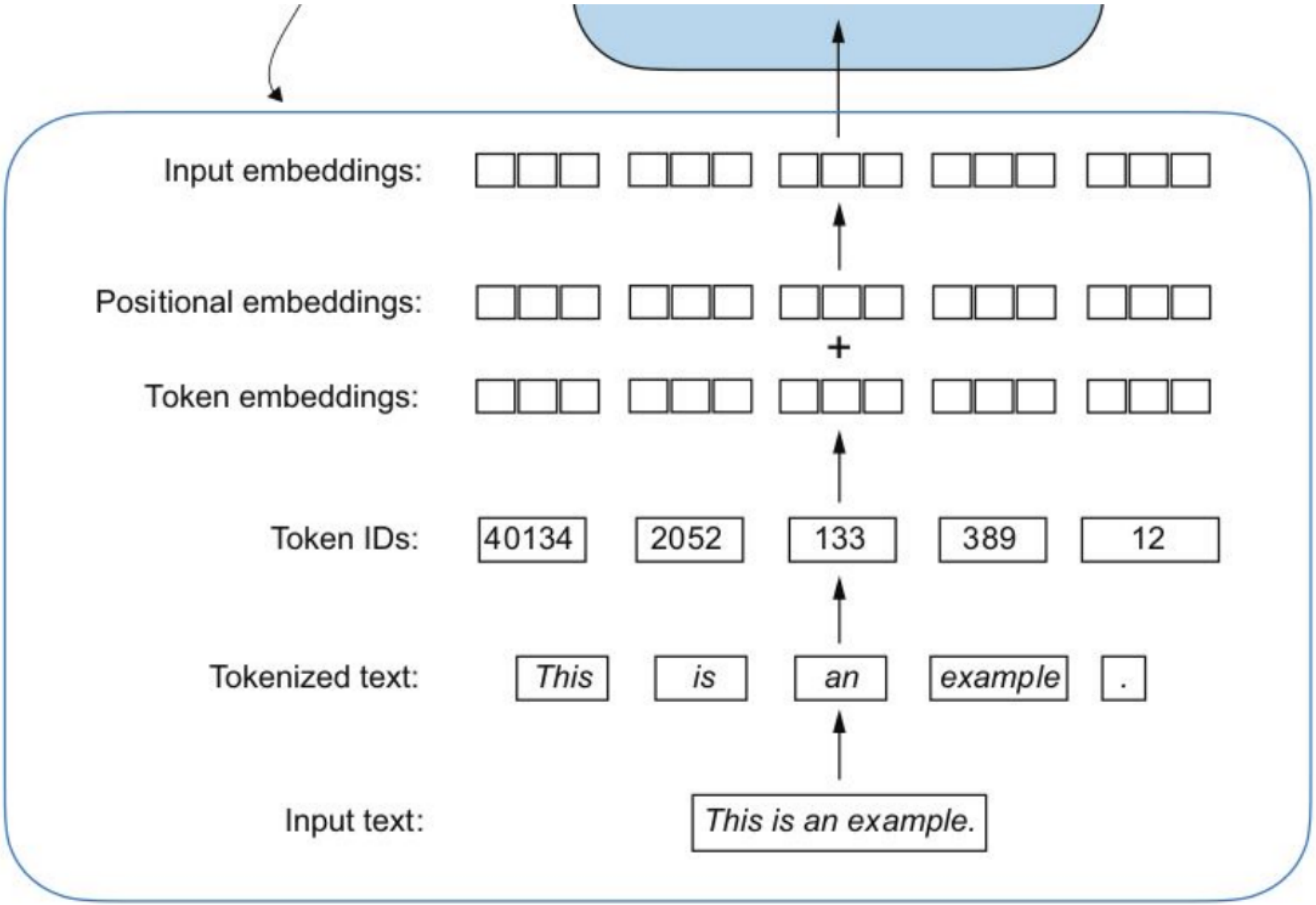


From text to vectors



Transformers





Input embeddings:
Positional embeddings:
+
Token embeddings:
Token IDs:
40134
2052
133
389
12
Tokenized text:
This
is
an
example
.
Input text:
This is an example.

Statistics

The smallest GPT-2 models (117M and 125M parameters) use an embedding size of 768 dimensions.

The largest GPT-3 model (175B parameters) uses an embedding size of 12,288 dimensions.

A self-attention head

Input: an embedding vector x(i) for each token i

Output: a context vector z(i) for each token i

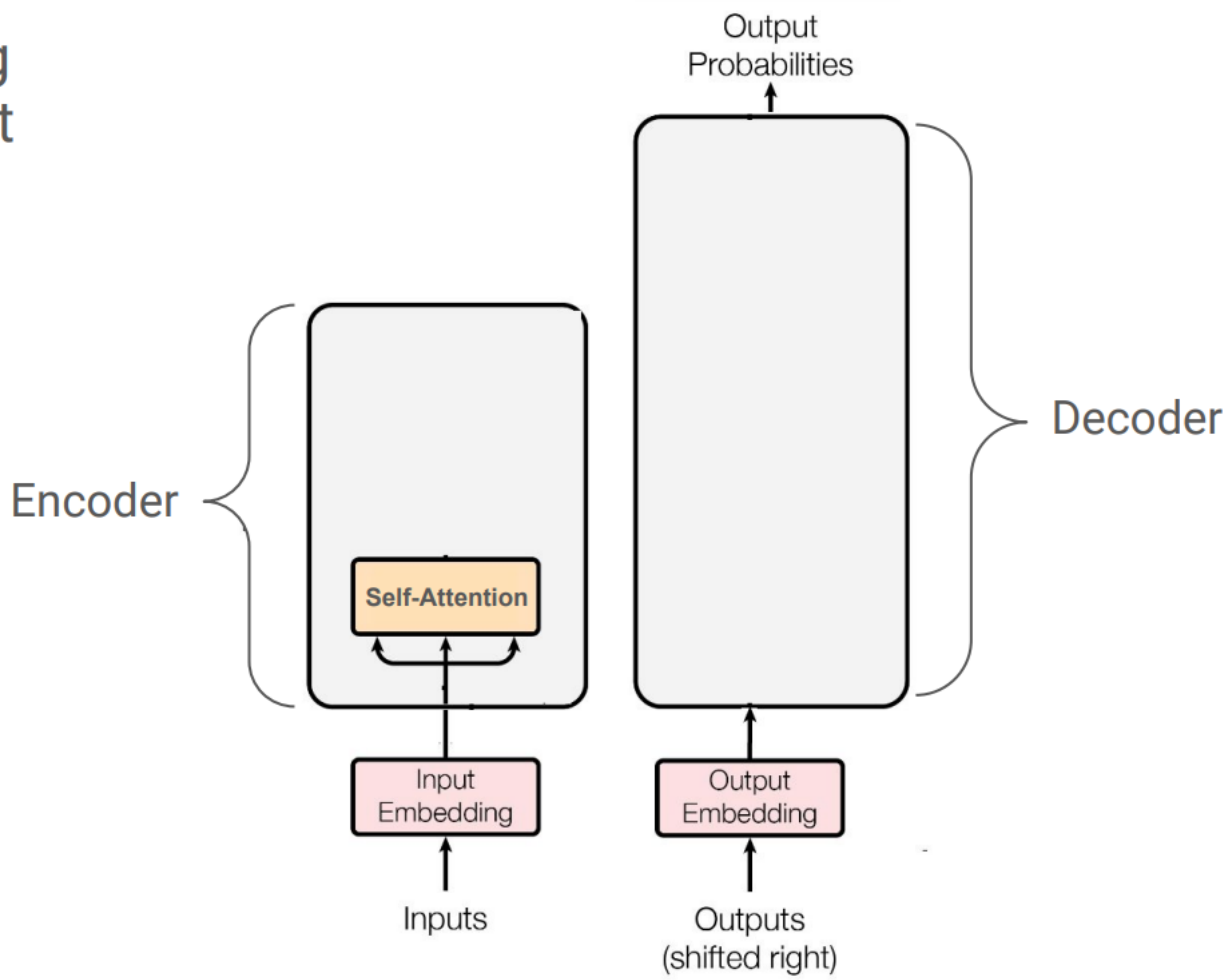
Intuition: z(i) gathers *contextual* information

Computing context vectors

Computing context vectors is very easy assuming we have computed **attention weights**: alpha(i,j) describes the importance of token j for token i.

Encoder: Self-Attention

Self-Attention is the core building block of Transformer, so let's first focus on that!



Keys, queries, and values

Input: an embedding vector x(i) for each token i

Output: for each token i:

- A *query* vector q(i), describing the information token i is interested in,
- A *key* vector k(i), whose goal is to match the relevant queries for token i,
- A *value* vector v(i), describing the information contained by token i.

Recipe for (Vectorized) Self-Attention in the Transformer Encoder

- Step 1: With embeddings stacked in X, calculate queries, keys, and values.

$$Q = XW^Q \qquad K = XW^K \qquad V = XW^V$$

- Step 2: Calculate attention score between query and keys.

$$E = QK^T$$

- Step 3: Take the softmax to normalize attention scores.

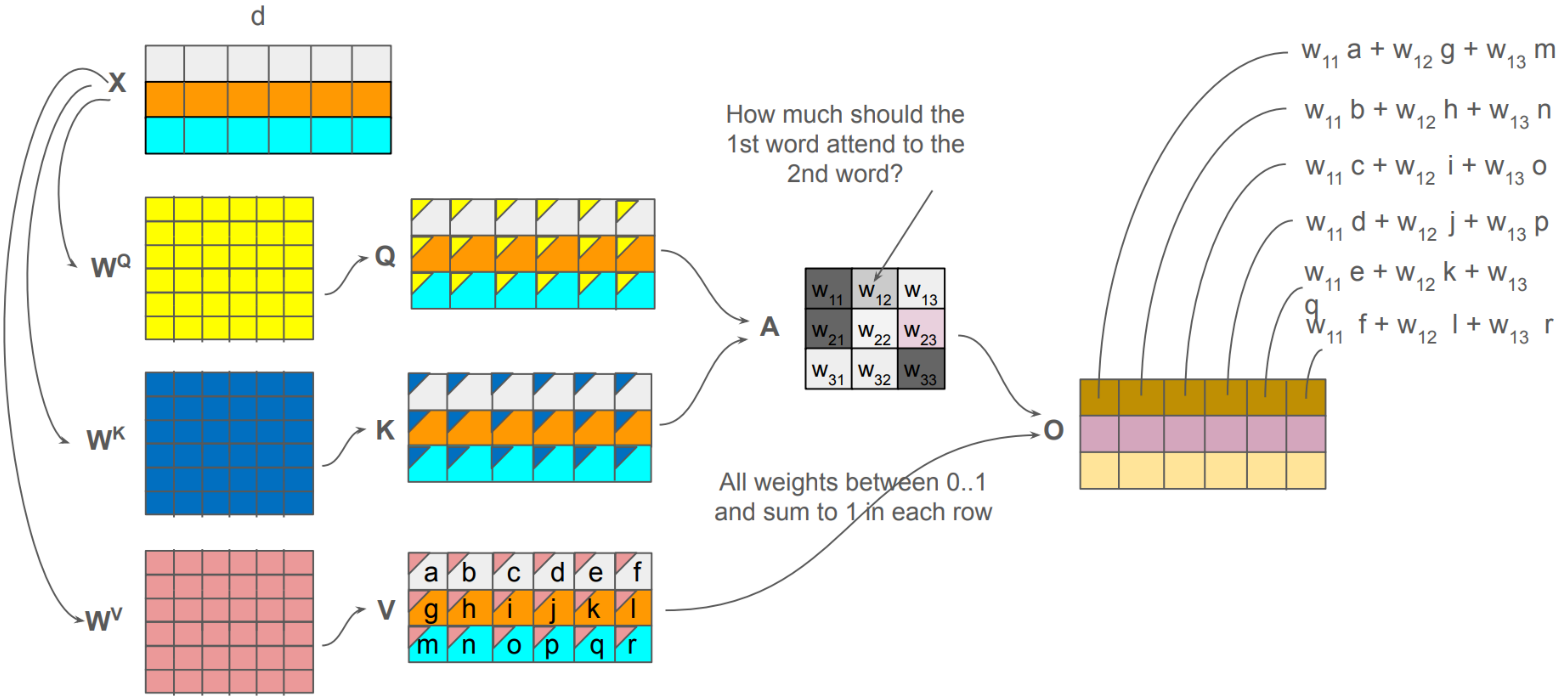
$$A = softmax(E)$$

- Step 4: Take a weighted sum of values.

$$Output = AV$$

$$\boxed{Output = softmax\left(QK^T\right)V}$$

In Pictures (N = 3, d = 6, h = 1)



Computing attention scores and weights

Now we focus on the core computation: attention scores and weights.

We first compute **attention scores**, and then normalise them into **attention weights**.