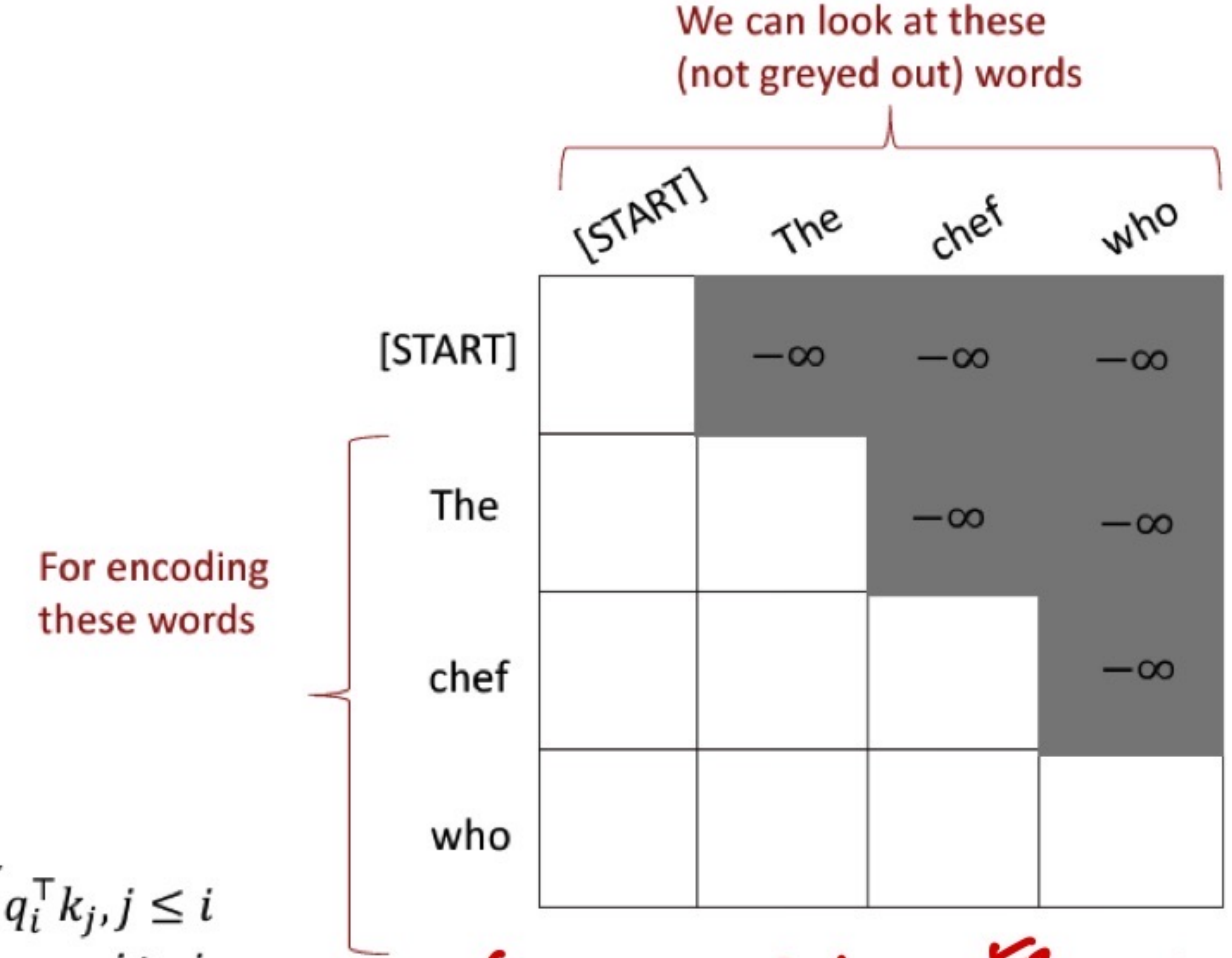
Masked Self-Attention

Masking the future in self-attention

- To use self-attention in **decoders**, we need to ensure we can't peek at the future.
- At every timestep, we could change the set of **keys and queries** to include only past words. (Inefficient!)
- To enable parallelization, we **mask out attention** to future words by setting attention scores to $-\infty$.

$$e_{ij} = \begin{cases} q_i^\top k_j, j \leq i \\ -\infty, j > i \end{cases}$$



$K_1 \leftarrow$ <START> $\rightarrow q_1$

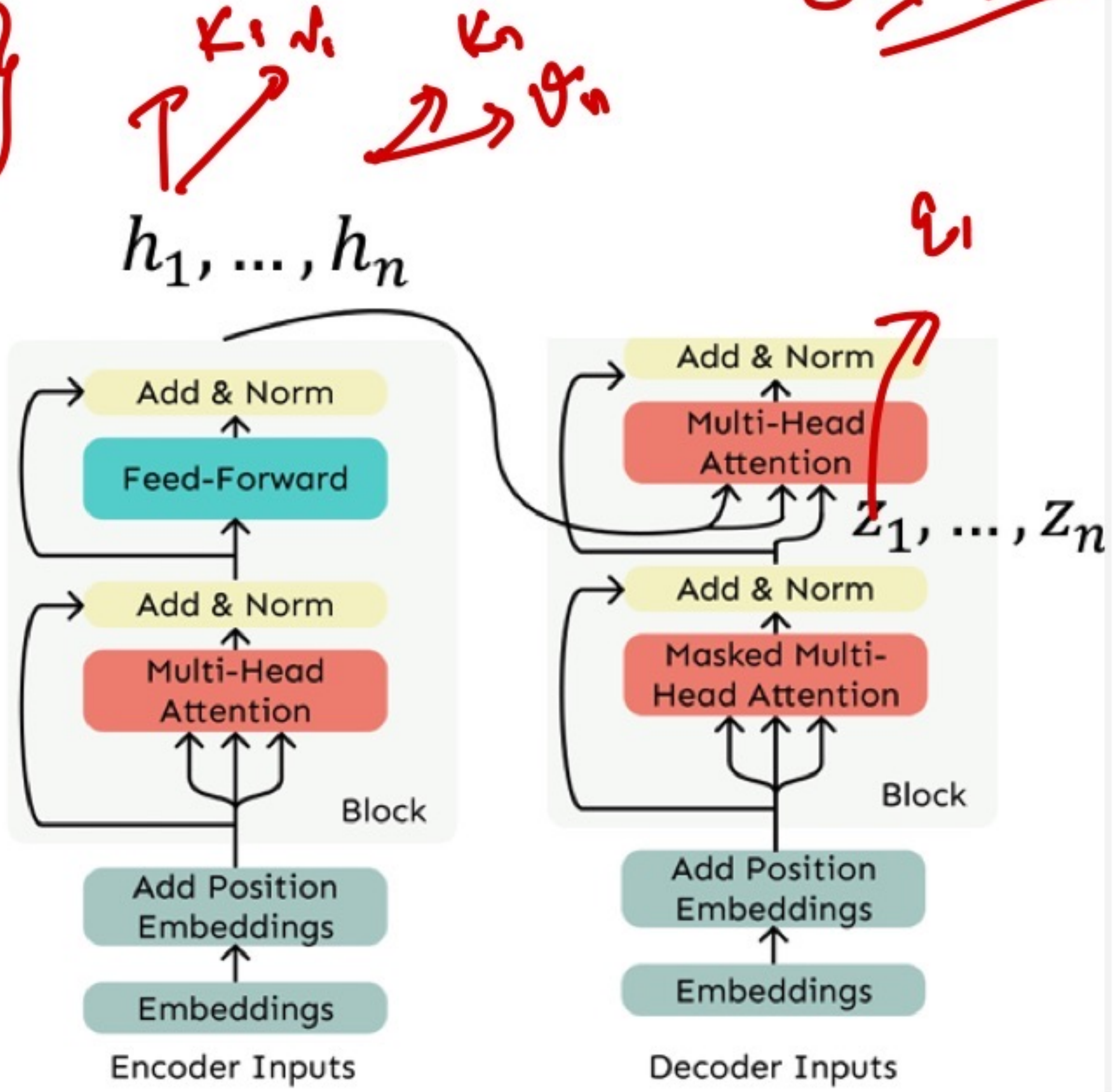
$K_2 \leftarrow$ The $\rightarrow q_2$

Encoder-Decoder Attention

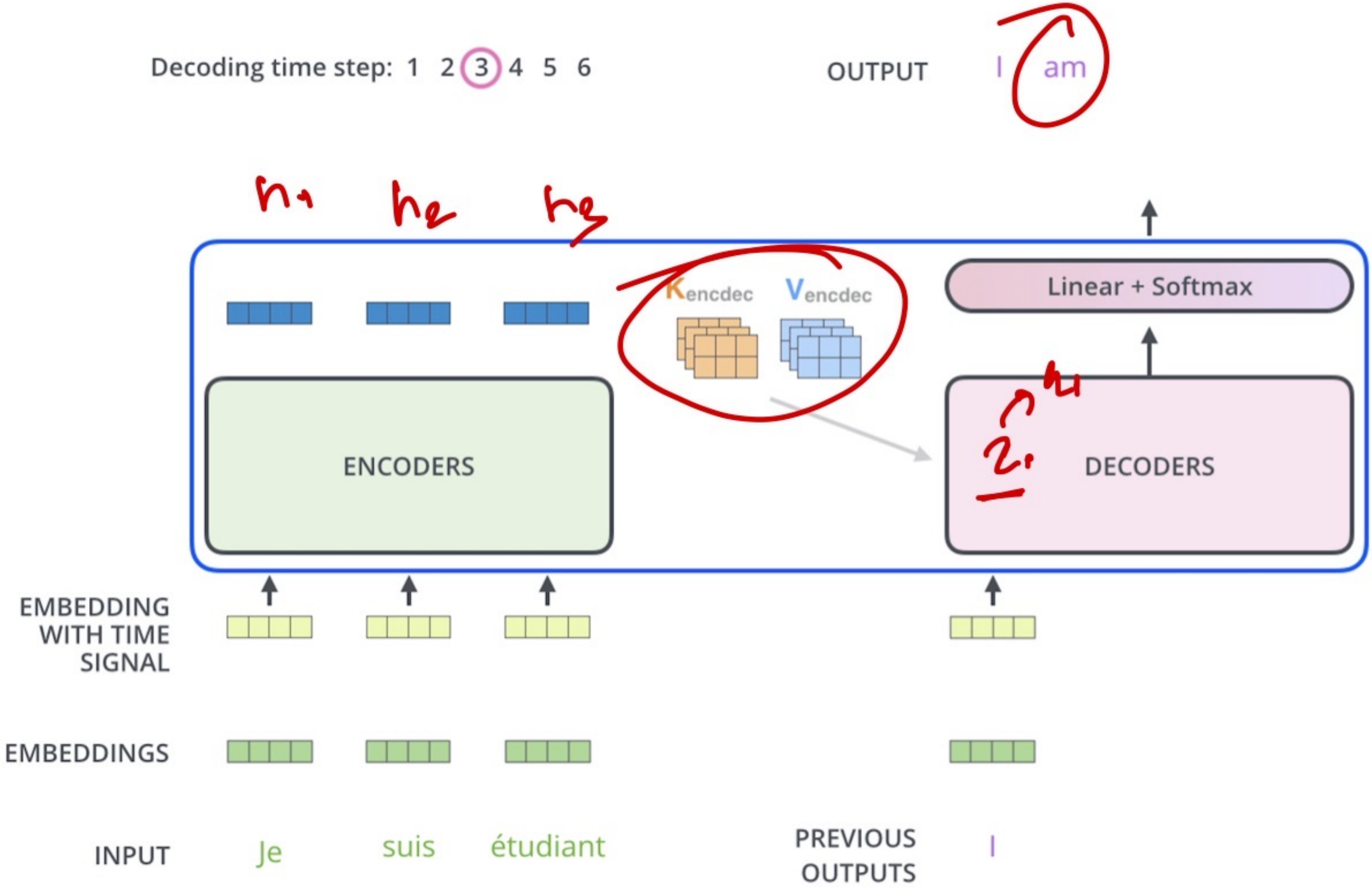
→ Cross-attention

Q, K, V

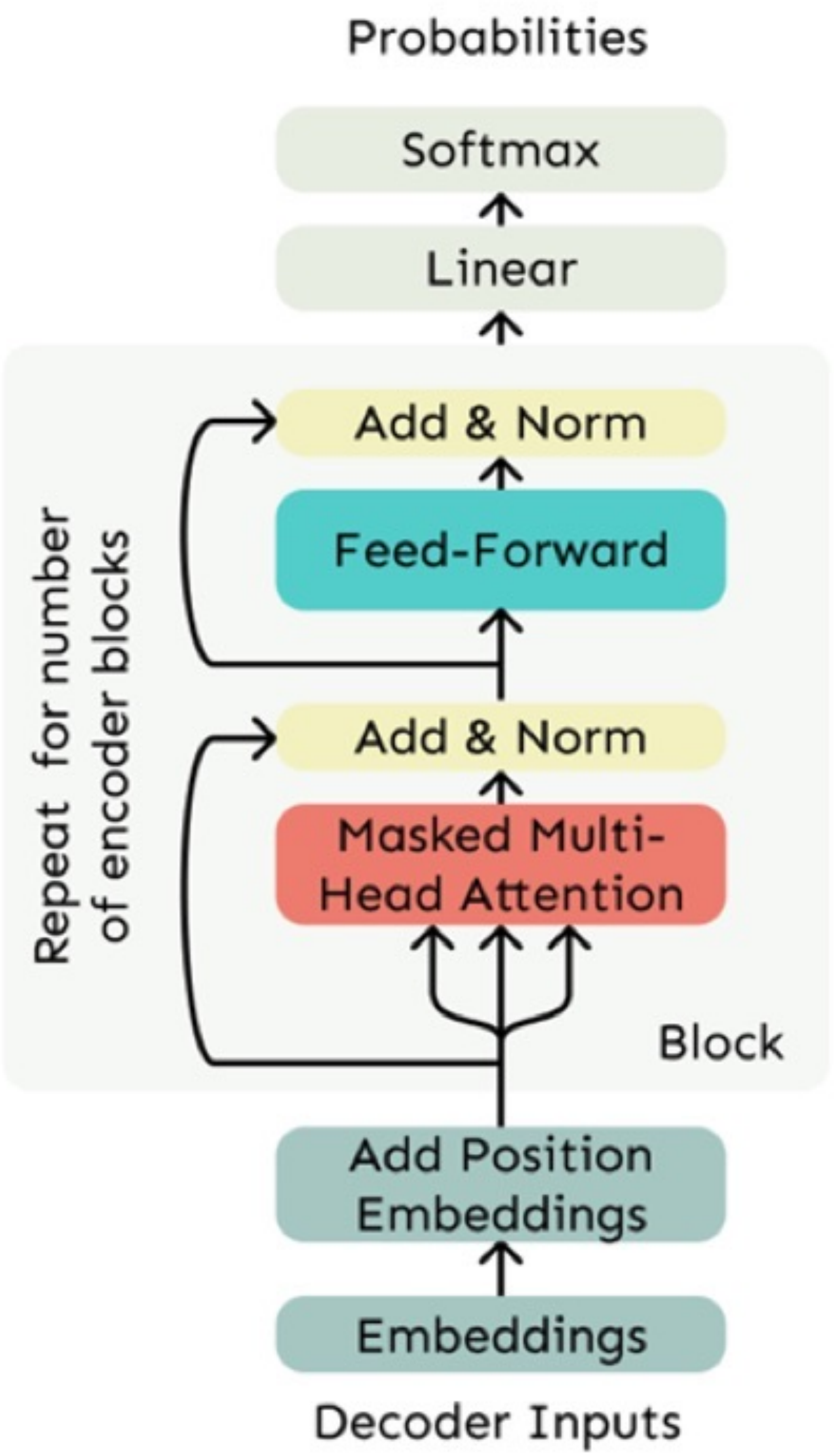
- We saw that self-attention is when keys, queries, and values come from the same source.
- In the decoder, we have attention that looks more like what we saw last week.
- Let $h_1, \ldots, h_n$ be **output** vectors **from** the Transformer **encoder**; $x_i \in \mathbb{R}^d$
- Let $z_1, \ldots, z_n$ be input vectors from the Transformer **decoder**, $z_i \in \mathbb{R}^d$
- Then keys and values are drawn from the **encoder** (like a memory):
  - $k_i = Kh_i, v_i = Vh_i$.
- And the queries are drawn from the **decoder**, $q_i = Qz_i$.



Encoder-Decoder Attention



Transformer as a Language Model (Decoder)



Unembedding layer in Transformer LM

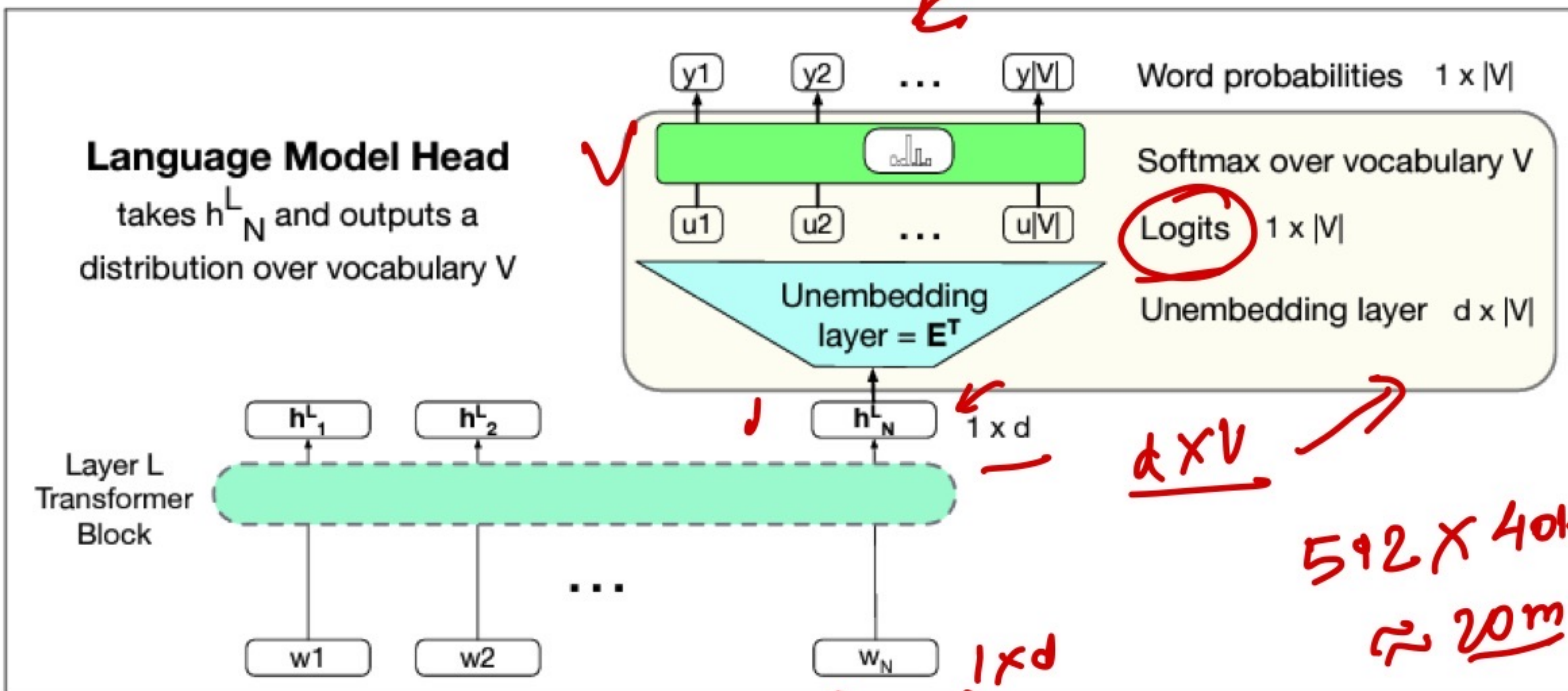


Figure 10.13 The language modeling head: the circuit at the top of a transformer that maps from the output embedding for token N from the last transformer layer ($\mathbf{h}_N^L$) to a probability distribution over words in the vocabulary V.

E

V x d

Weight Tying

- The unembedding layer can be learned, but it is also very common to tie this matrix to the embedding matrix E.
- At the input, embedding layer ($[E : V \times d]$) is used to map from one-hot (V-dim) to an embedding (d-dim).
- At the language modeling head, transpose of the embedding layer ($[E^T : d \times V]$) is used to map back from the embedding (d-dim) to a vector over the vocabulary (V-dim).
- In the learning process, E is optimized to be good at doing both of these mappings.

Text Completion via Language Models

Conditional Generation Task

The task of generating text conditioned on an input piece of text

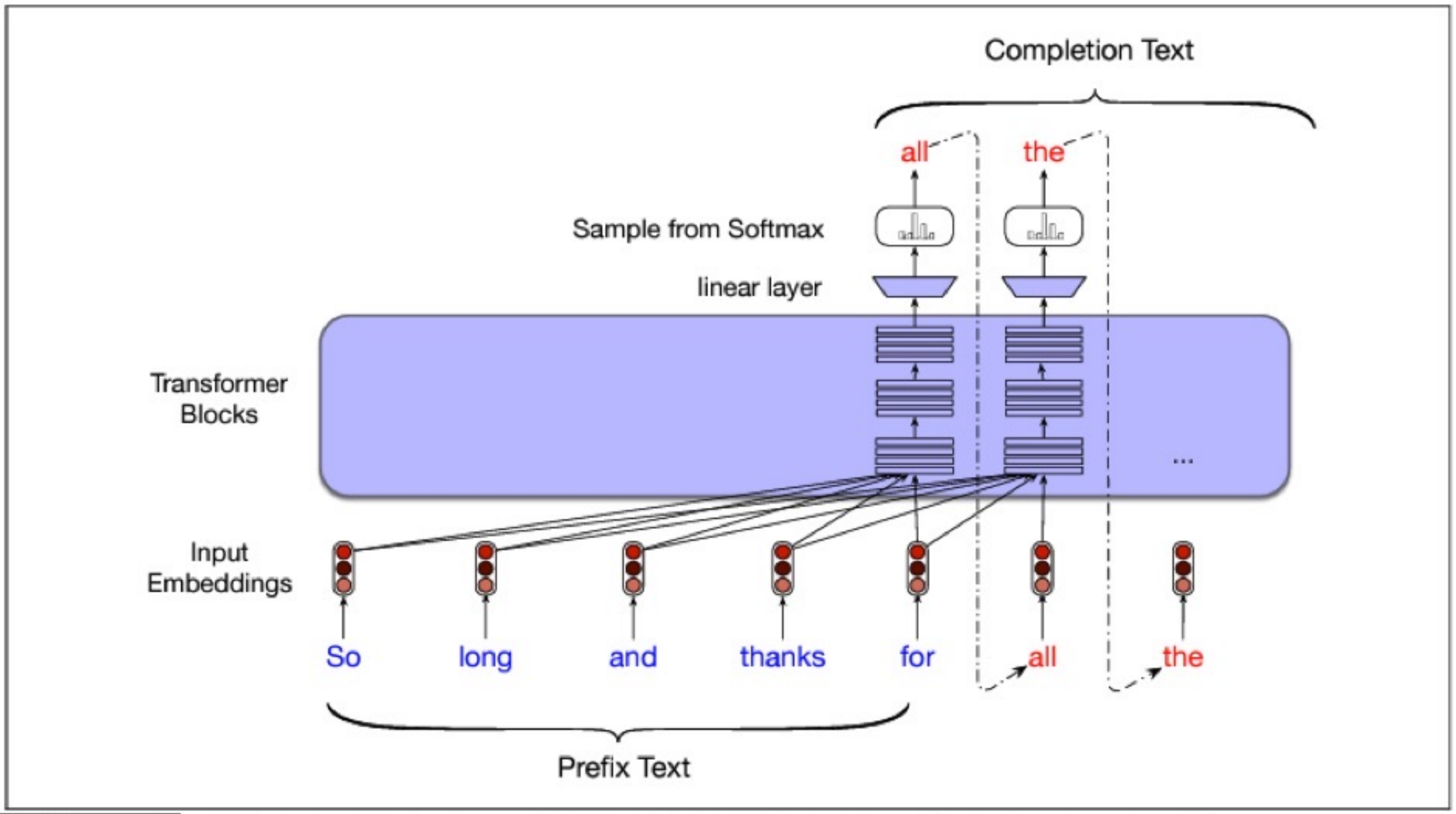


Figure 10.15 Autoregressive text completion with transformer-based large language models.

Large Language Models: Main Insight

prefix

- LM is given a test ~~suffix~~ (context) and is asked to generate a possible completion
- As the generation proceeds, model has access to the context as well all of its previously generated tokens
- This ability is the key to the power of Large Language Models built from transformers

But why do we care about predicting next words?

Many practical NLP tasks can be cast as word prediction / text completion