



Language Understanding

05 - Transformers

Hossein Zeinali

Introduction

- Assume we have a fixed size sequence, so sequence modeling is a problem of:

$$f: \mathbb{R}^d \rightarrow \mathbb{R}$$

where the input is a fixed size vector.

- But, documents have variable lengths. We should somehow encode a document in a fixed size vector.
- Typical ways to do this:



Introduction

- Bag of Words:

- One dimension per word in vocabulary ($d \approx 100,000$)
- Almost all values are zero
 - Can use sparse data and only store non-zero data
- Cannot preserve word order while order matters in our tasks.
 - E.g. “work to live” vs. “live to work”
- N-grams as a solution:
 - The dimensionality is d^N

- RNN:

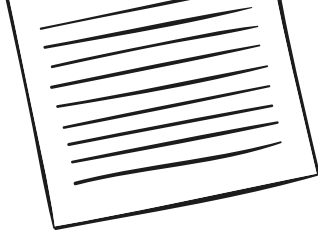
- We saw how we can use RNN for sequence modeling
- Main problem: Vanishing & Exploding Gradients
- It is practical for only very short segments



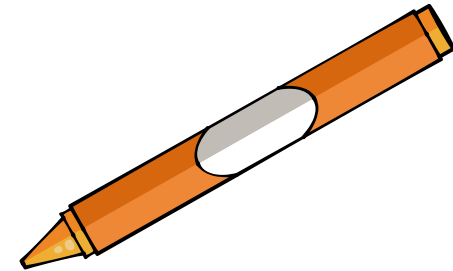
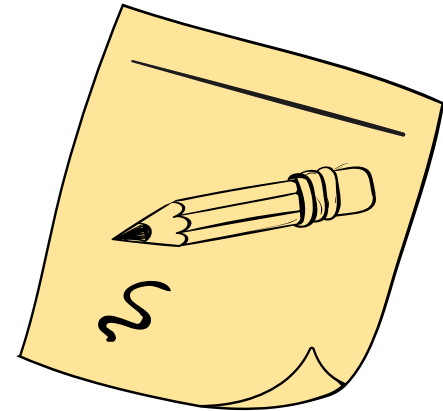
Introduction

- LSTM:
 - Again, we saw how LSTM can be used for sequence modeling
 - Difficult to train
 - Forward pass is not parallelizable like RNN
 - Very long gradients paths for long sequences
 - Vanishing gradient, even with a forget gate
 - Transfer learning never really worked
 - Finetuning a pre-trained network with small labeled training data
 - It needs lots of labeled data to train the network from scratch
- Seq2seq models with attentions:
 - Has better performance because it uses a combination of encoder hidden states.
 - Allows modeling of dependencies without regard to their distance in the input or output sequences
 - But, in all but a few cases, encoder and decoder are still RNN/LSTM.

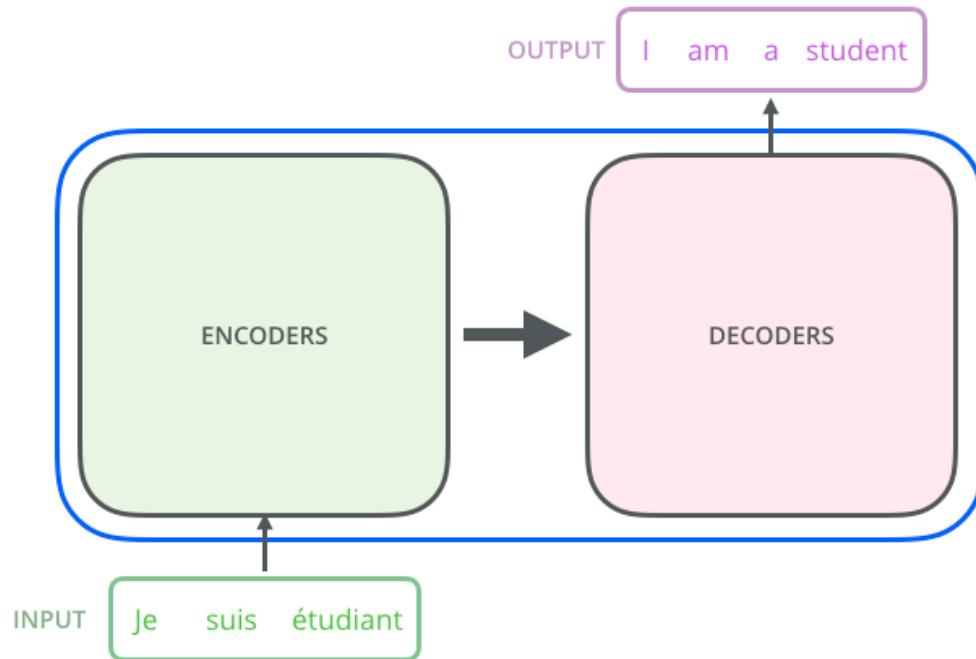




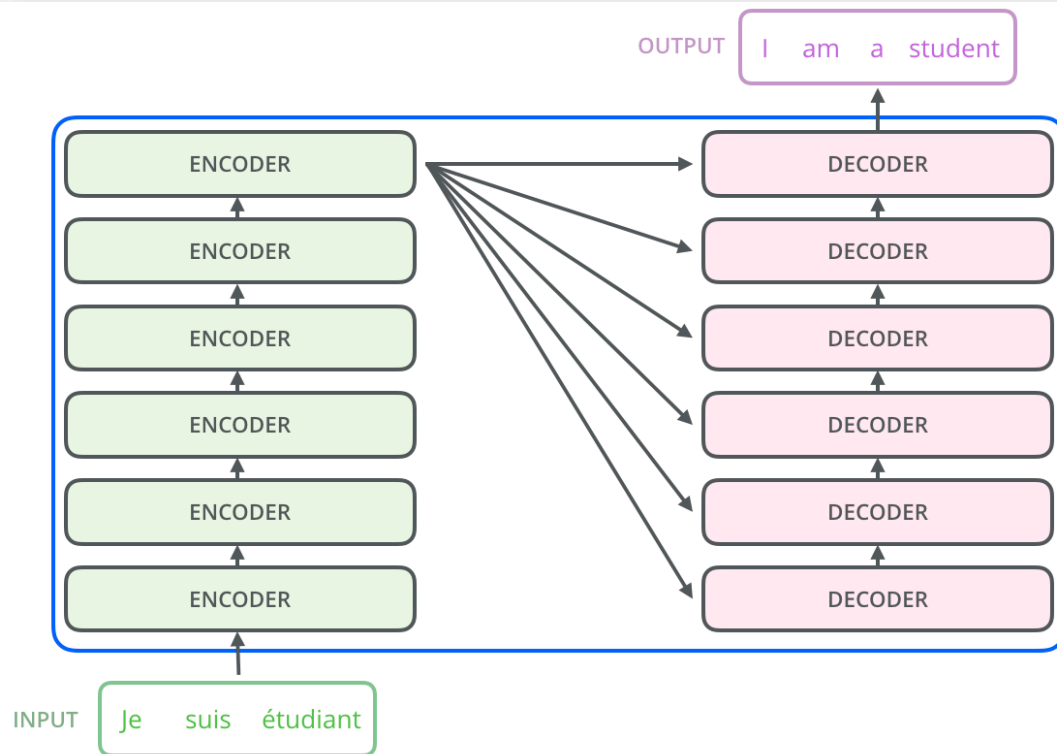
Transformer



Transformer Topology

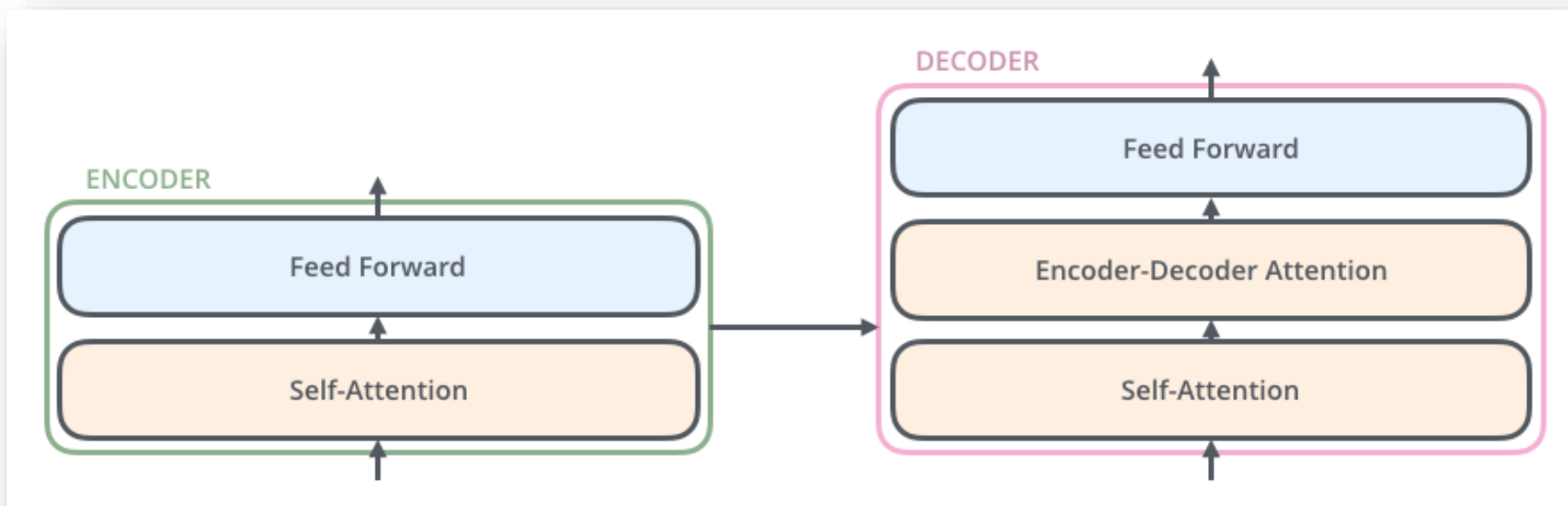


Transformer Topology

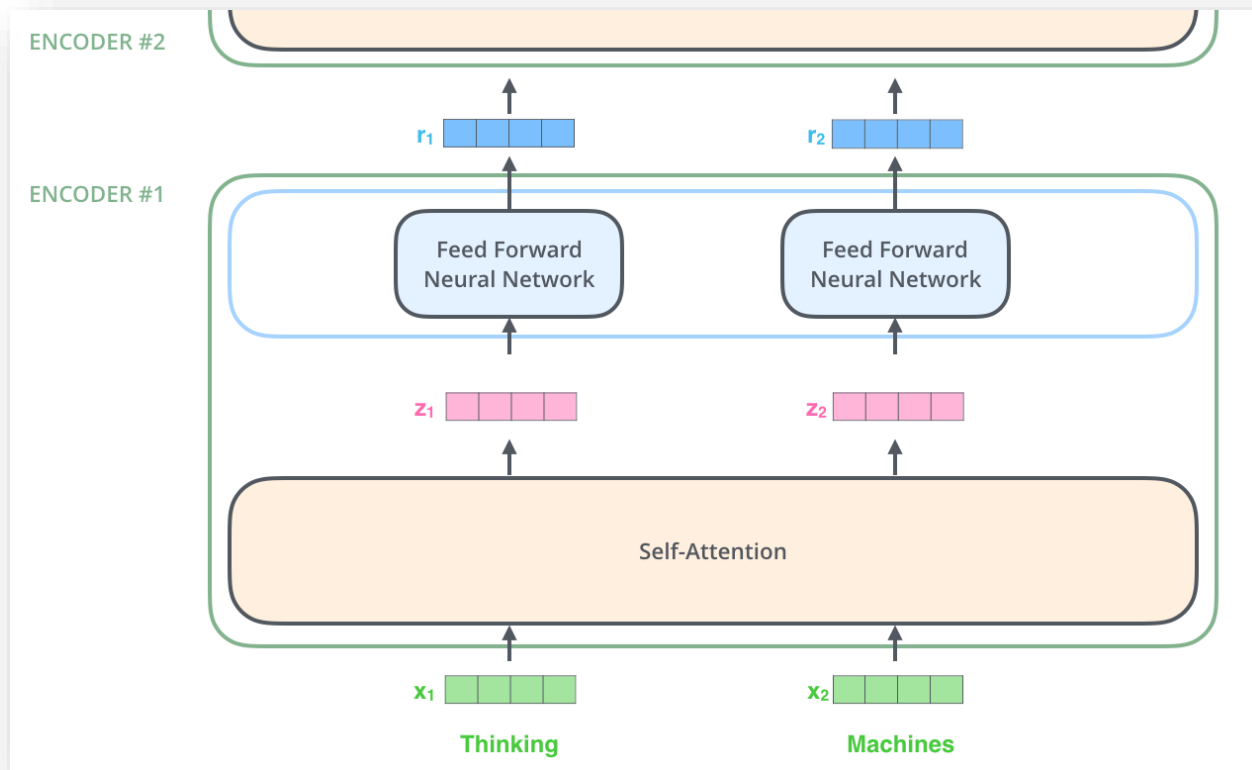


Transformer Topology

- The encoders (decoders) are all identical in structure, but they do not share weights.

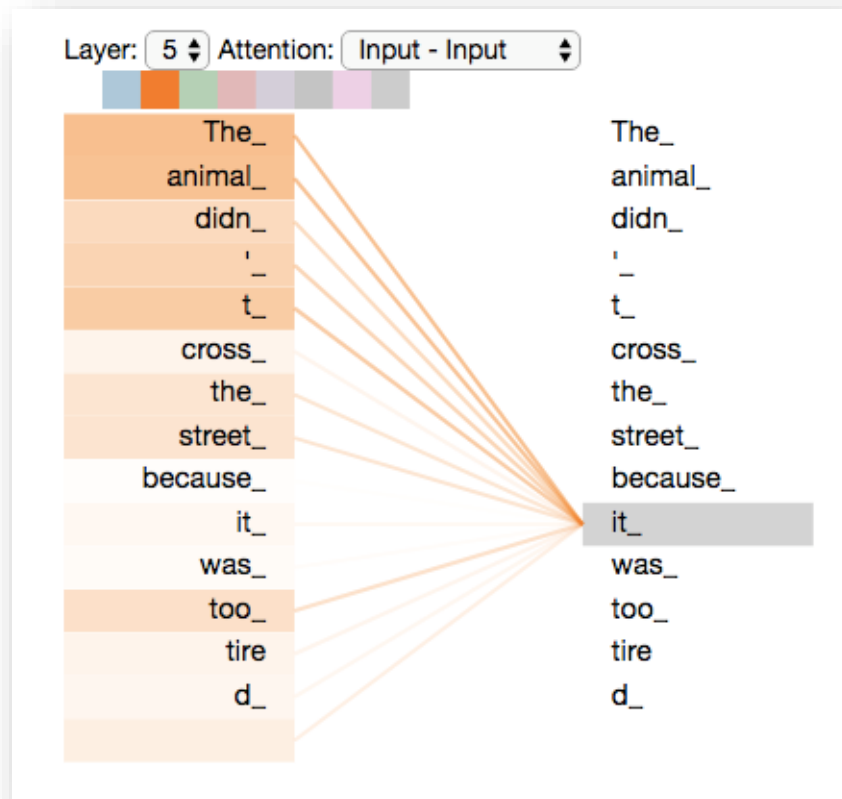


Encoder Topology

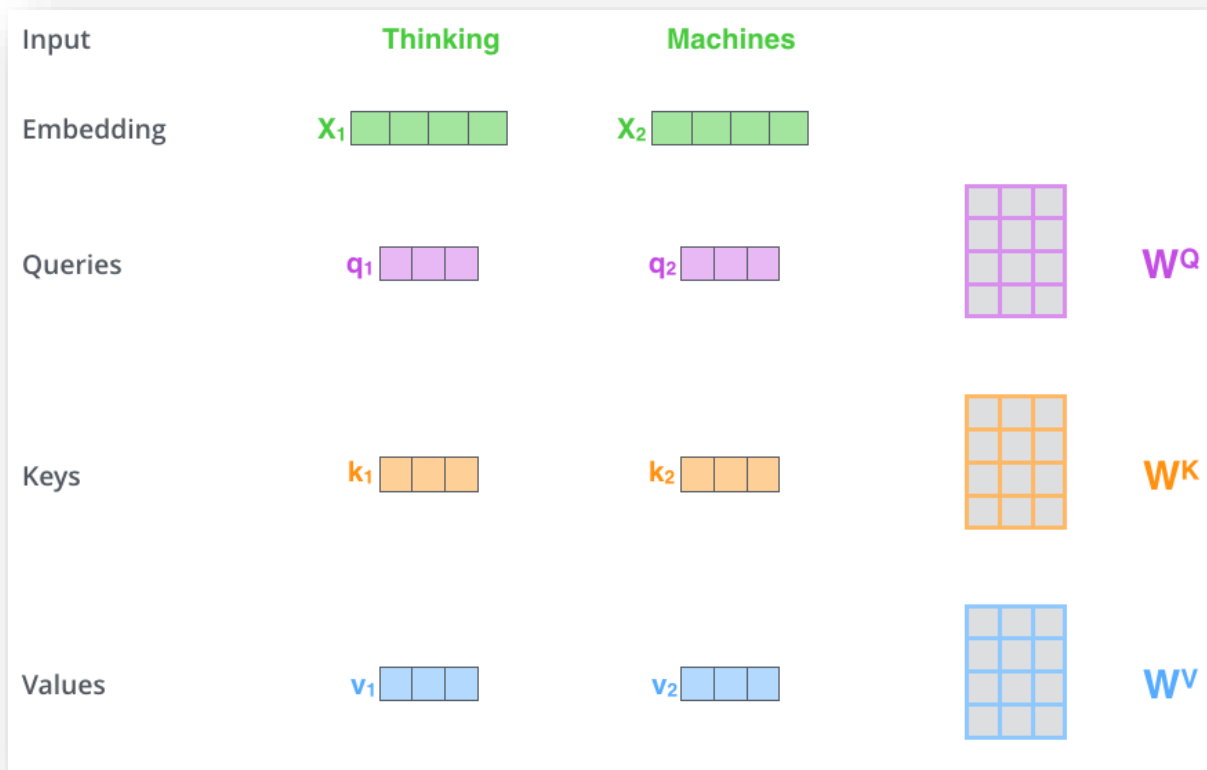


Self-Attention Layer

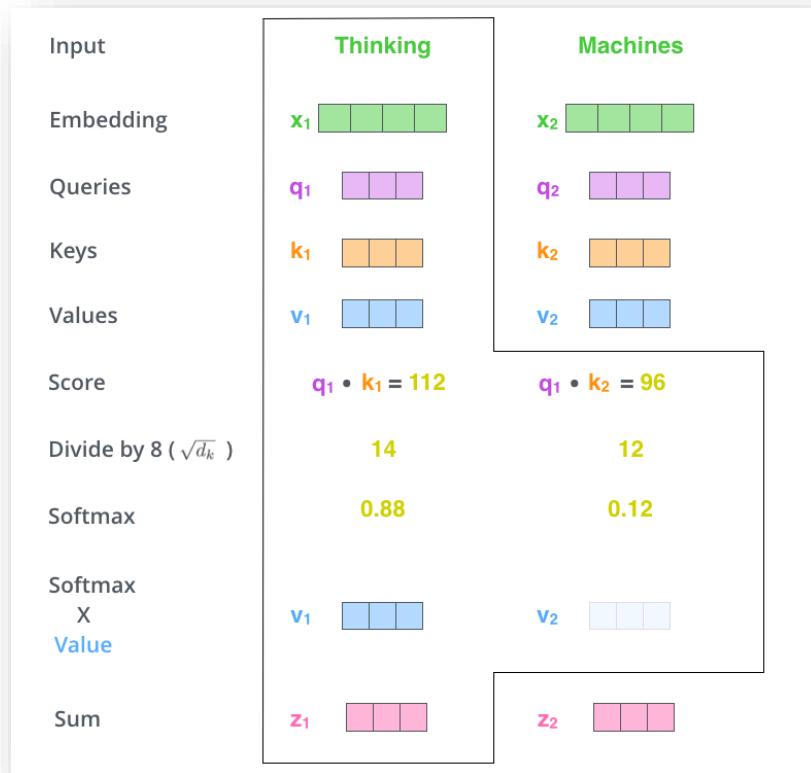
- Input: “The animal didn't cross the street because it was too tired”



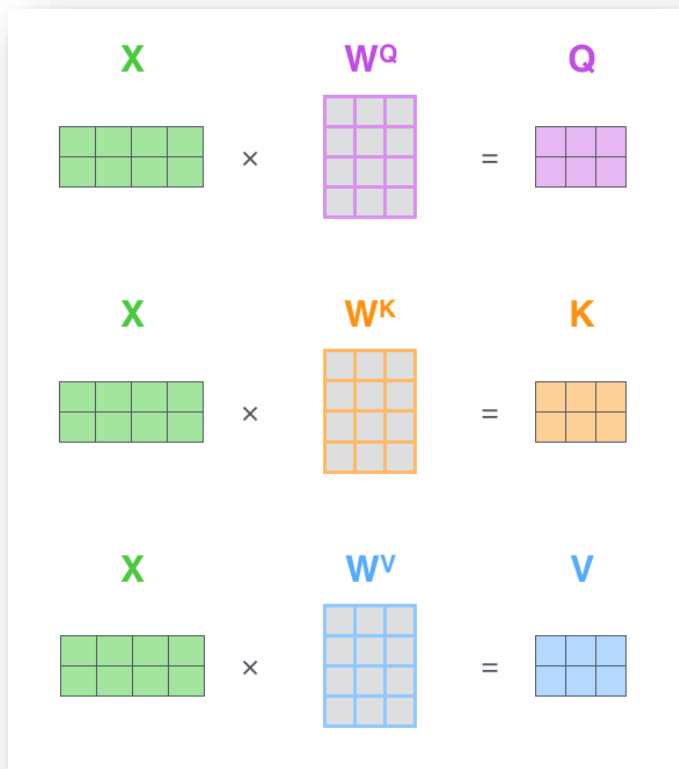
Self-Attention Layer



Scaled Dot-Product Attention

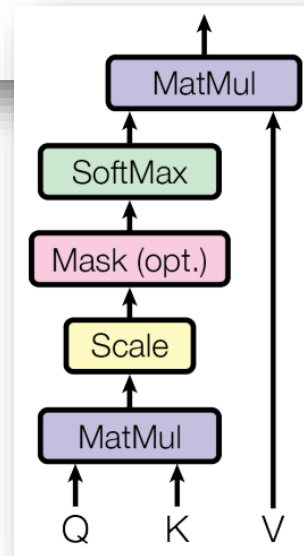


Scaled Dot-Product Attention



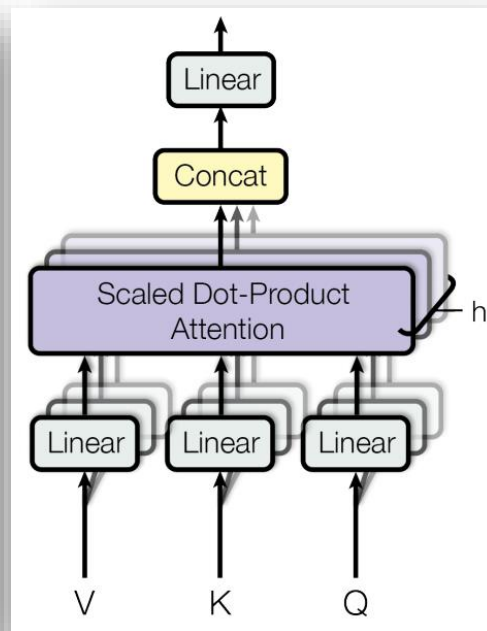
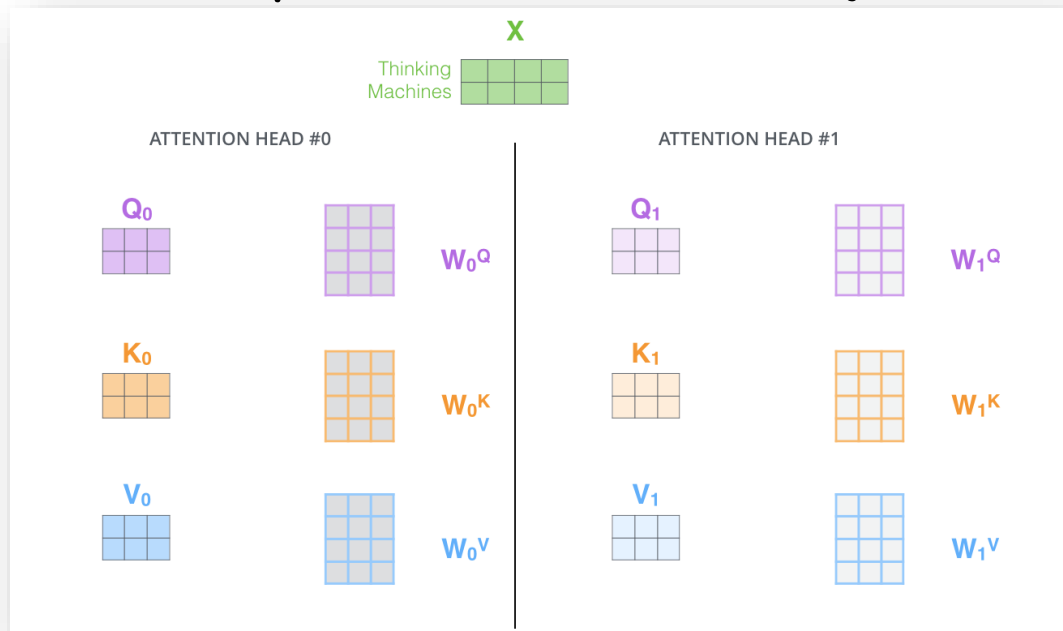
$$\text{softmax}\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) = Z$$

The diagram shows the softmax operation applied to the scaled dot product of Q and K^T . The result is a 2x2 matrix Z (pink).

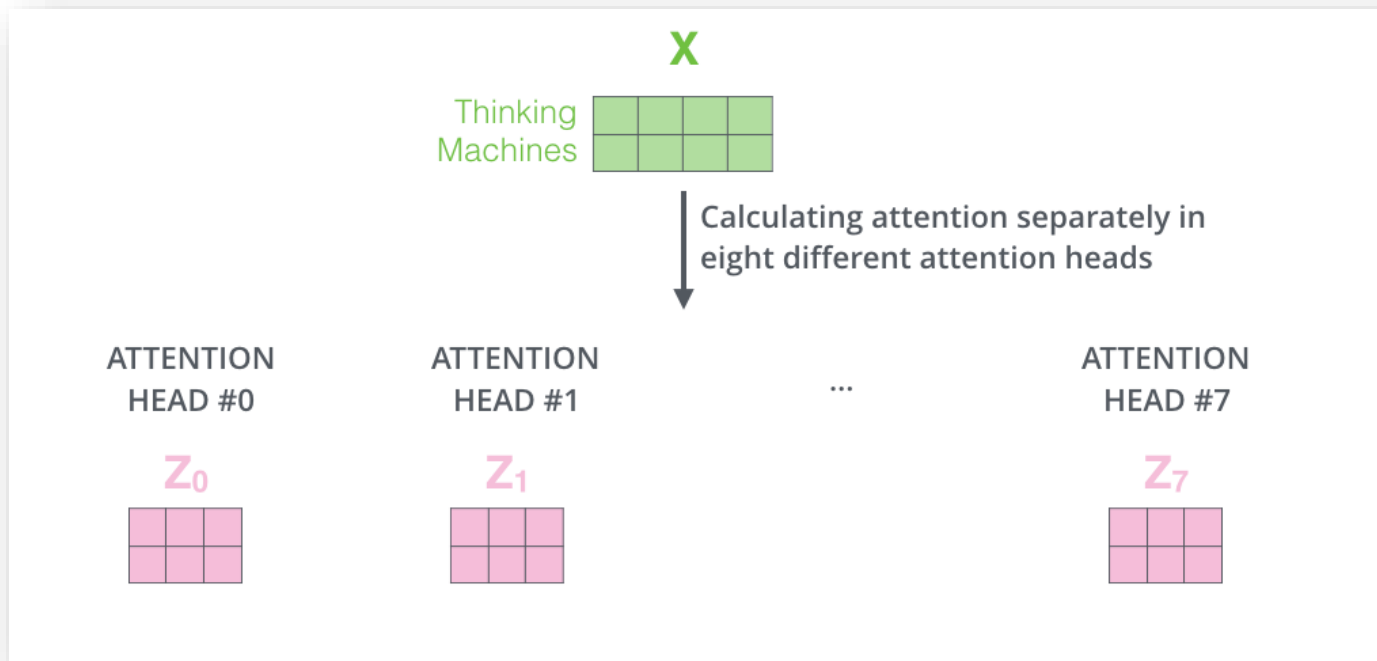


Multi-Head Attention

- Multi-head attention expands the model's ability to focus on different positions simultaneously.



Multi-Head Attention



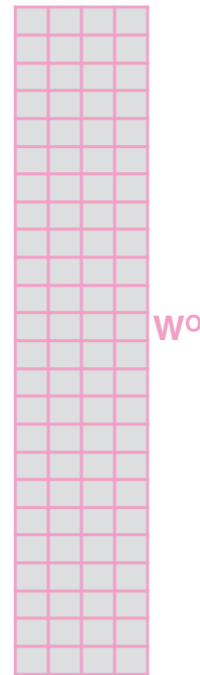
Multi-Head Attention

1) Concatenate all the attention heads



2) Multiply with a weight matrix W^O that was trained jointly with the model

X



3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN



Multi-Head Attention: All-in-One!

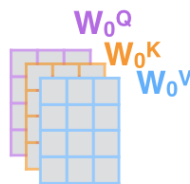
1) This is our input sentence*

Thinking
Machines

2) We embed each word*



3) Split into 8 heads. We multiply X or R with weight matrices



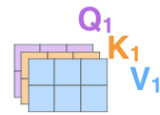
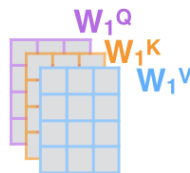
4) Calculate attention using the resulting $Q/K/V$ matrices



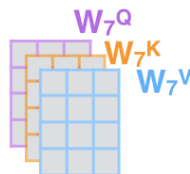
5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer



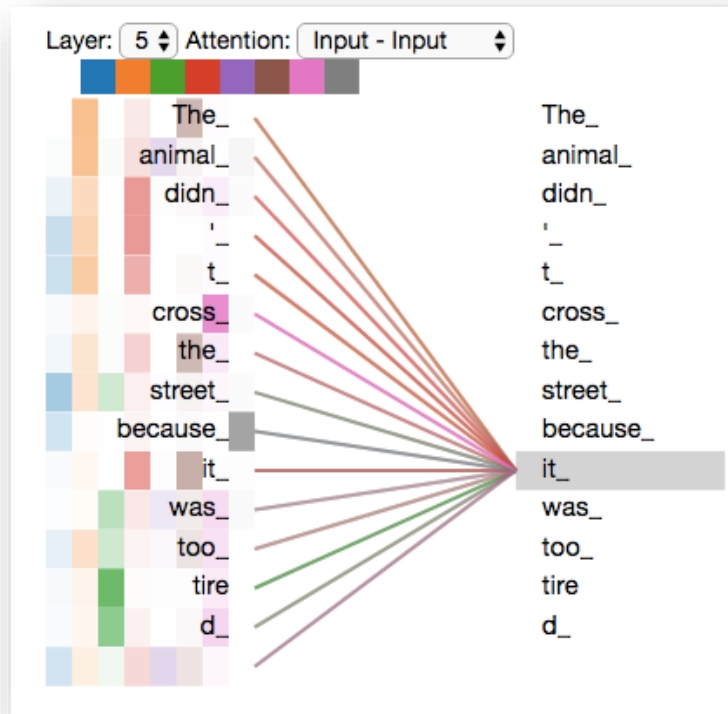
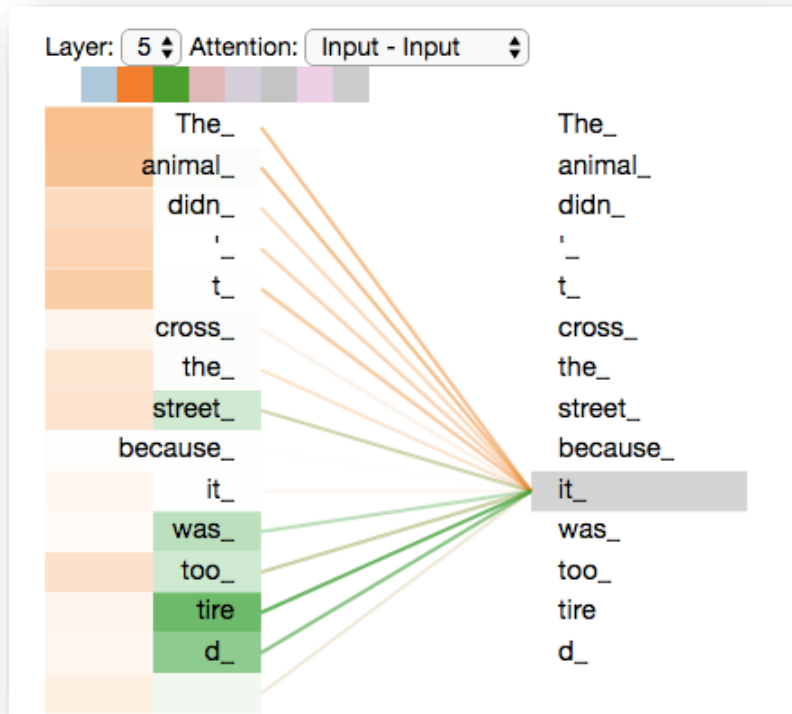
* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one



...



Multi-Head Attention



Position-wise Feed-Forward Networks

- There is a fully connected feed-forward network (FFN) in each layer of encoder and decoder, which is applied to each position separately and identically.
- This FFN consists of two linear transformations with a ReLU activation in between:

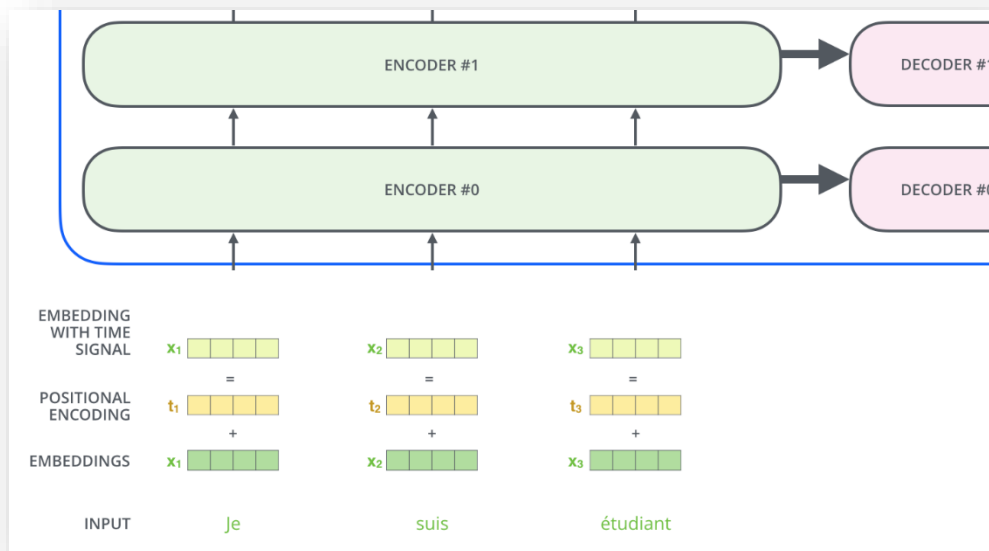
$$FFN(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

- The dimensionality of input and output is $d_{model} = 512$, and the inner-layer has dimensionality $d_{FF} = 2048$.



Positional Encoding

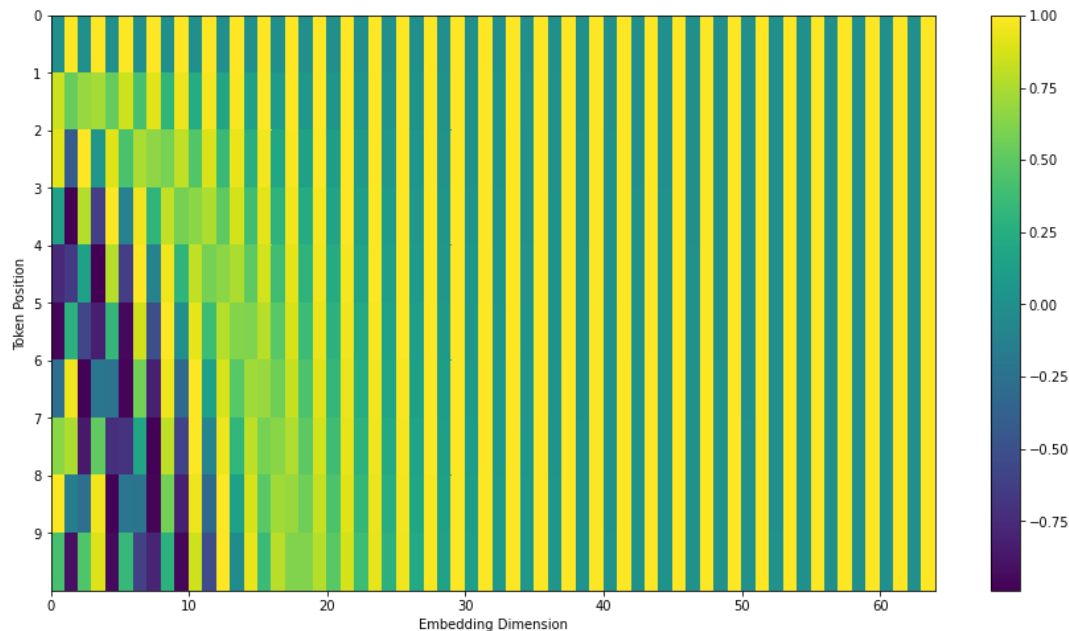
- To preserve the order of the words in the input sequence, the transformer adds a vector to each input embedding.
 - Determine the position of each word, or the distance between different words



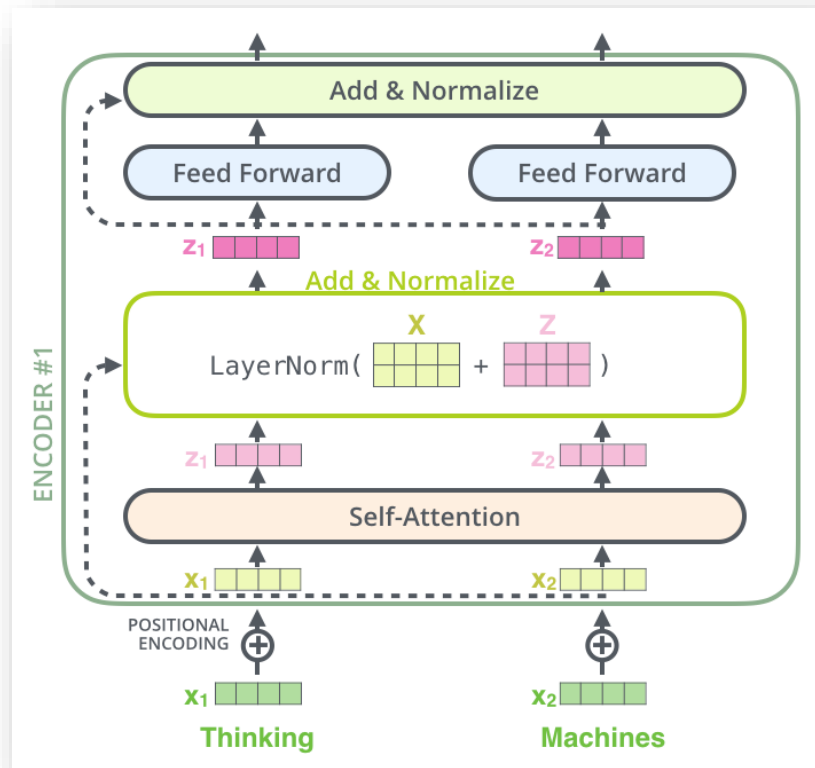
Positional Encoding

$$PE_{(pos, 2i)} = \sin(pos/10000^{2i/d_{\text{model}}})$$

$$PE_{(pos, 2i+1)} = \cos(pos/10000^{2i/d_{\text{model}}})$$



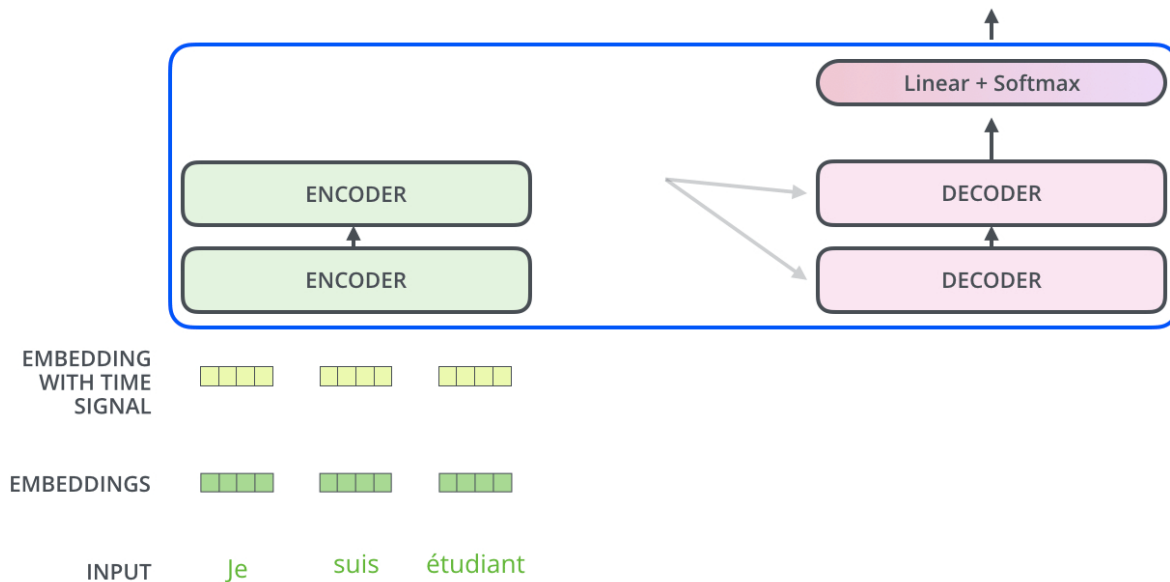
The Residuals and Layer-norm



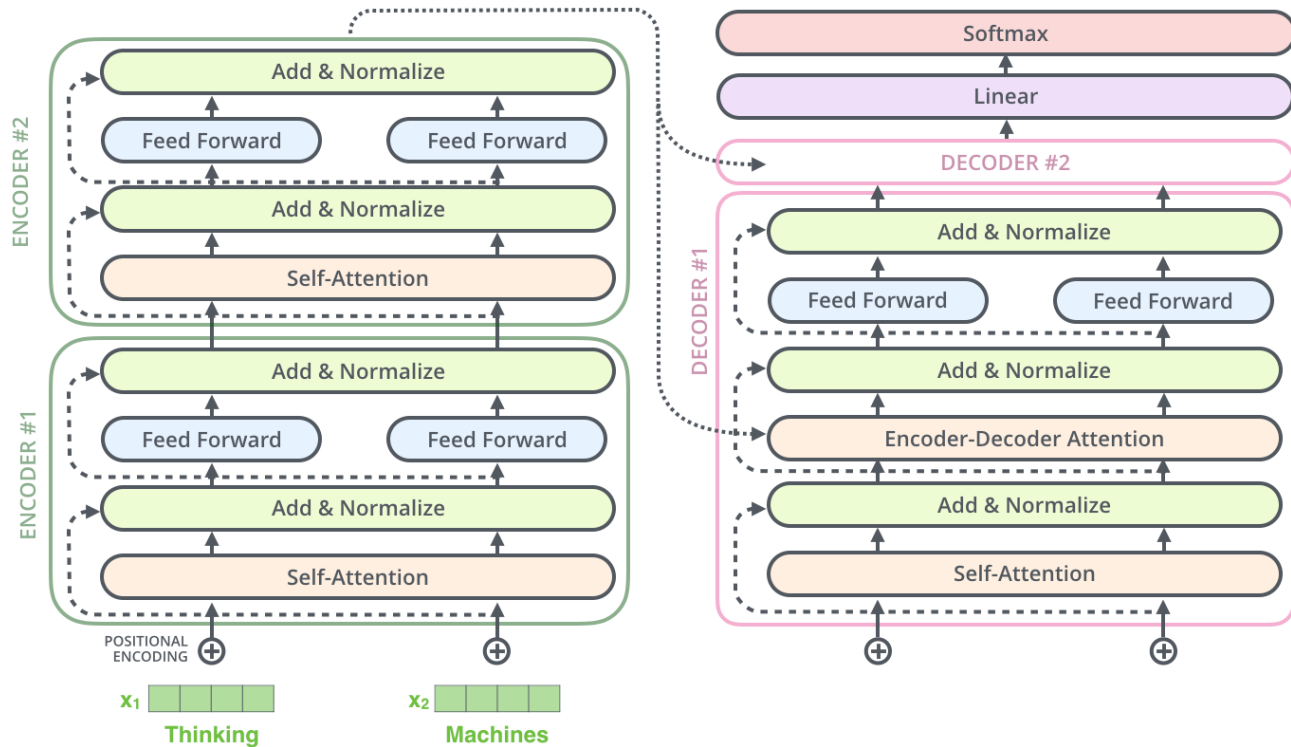
The Decoder Side

Decoding time step: 1 2 3 4 5 6

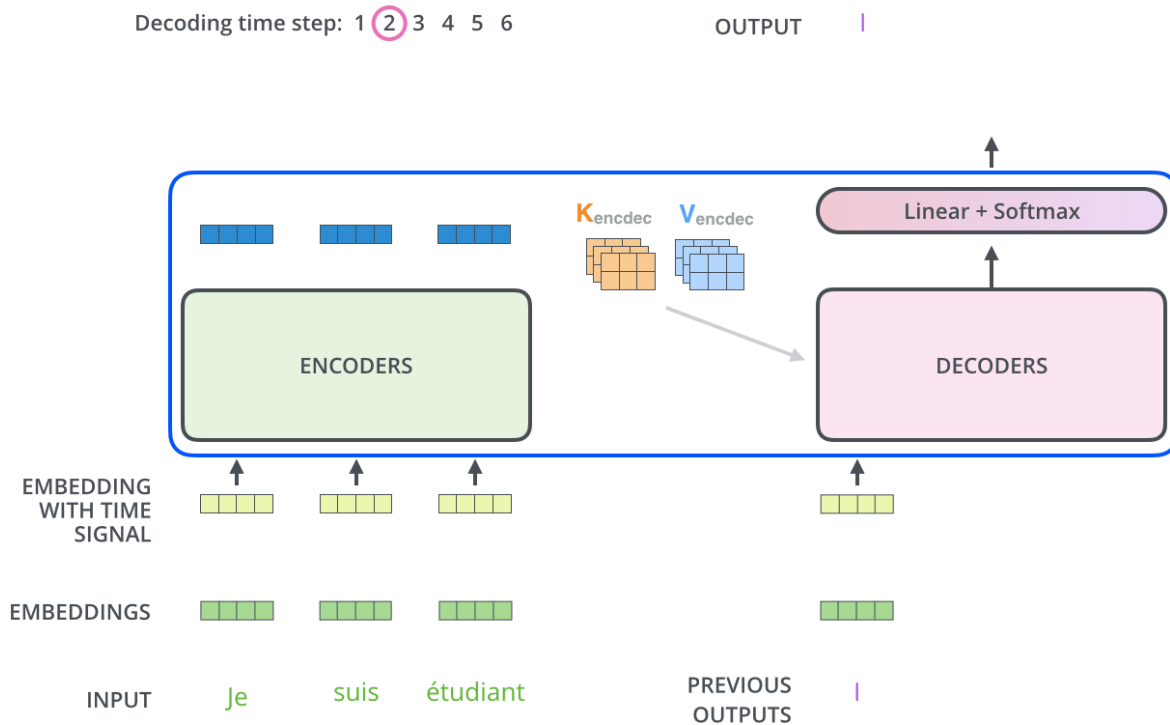
OUTPUT



The Decoder Side



The Decoder Side



The Final Linear and Softmax Layer

Which word in our vocabulary
is associated with this index?

am

Get the index of the cell
with the highest value
(argmax)

5

log_probs



Softmax

logits



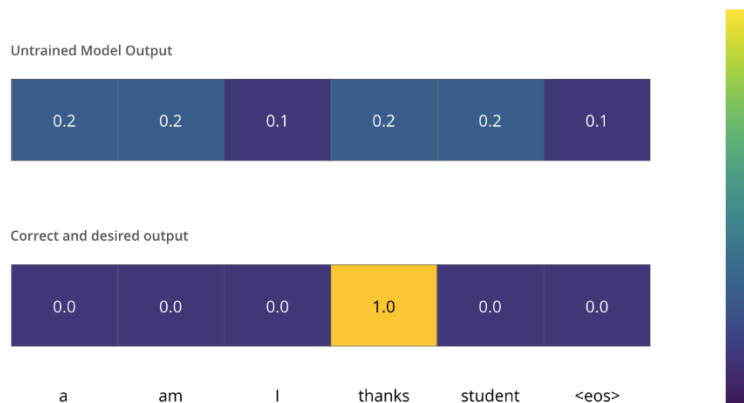
Linear

Decoder stack output



The Loss Function and Training

- One-hot encoding is used to encode the output vocabulary
- Cross-entropy loss is used for training the network

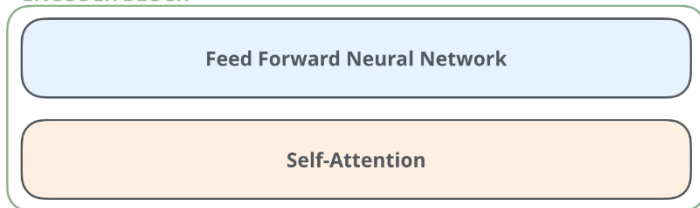


The Decoder and Encoder Blocks



THE TRANSFORMER

ENCODER BLOCK

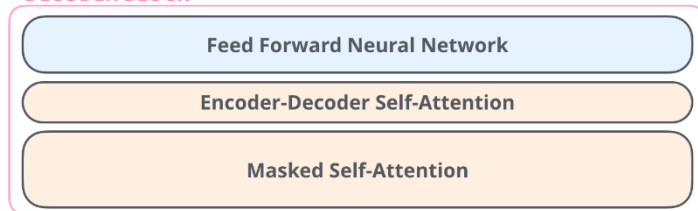


robot	must	obey	orders	<eos>	<pad>	...	<pad>
1	2	3	4	5	6		512



THE TRANSFORMER

DECODER BLOCK

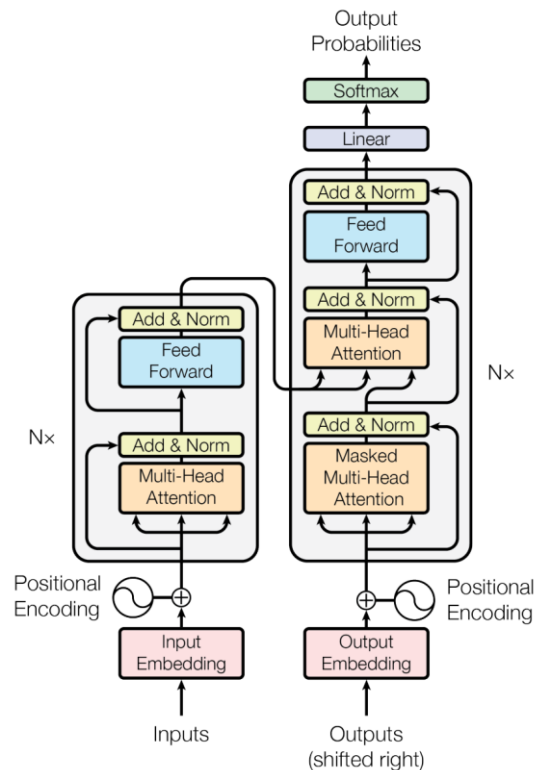


Input

<s>	robot	must	obey				
1	2	3	4	5	6		512



The Whole Transformer Model



New Related Papers

- Child, Rewon, et al. "Generating long sequences with sparse transformers." *arXiv preprint arXiv:1904.10509* (2019).
- Kitaev, Nikita, Łukasz Kaiser, and Anselm Levskaya. "Reformer: The efficient transformer." *arXiv preprint arXiv:2001.04451* (2020).
- Making Transformer networks simpler and more efficient
 - <https://ai.facebook.com/blog/making-transformer-networks-simpler-and-more-efficient/>





Thanks for your attention



References and IP Notice

- [1] Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems*. 2017.
- Most of the figures are selected from <https://jalammar.github.io> web site.

