Transformers

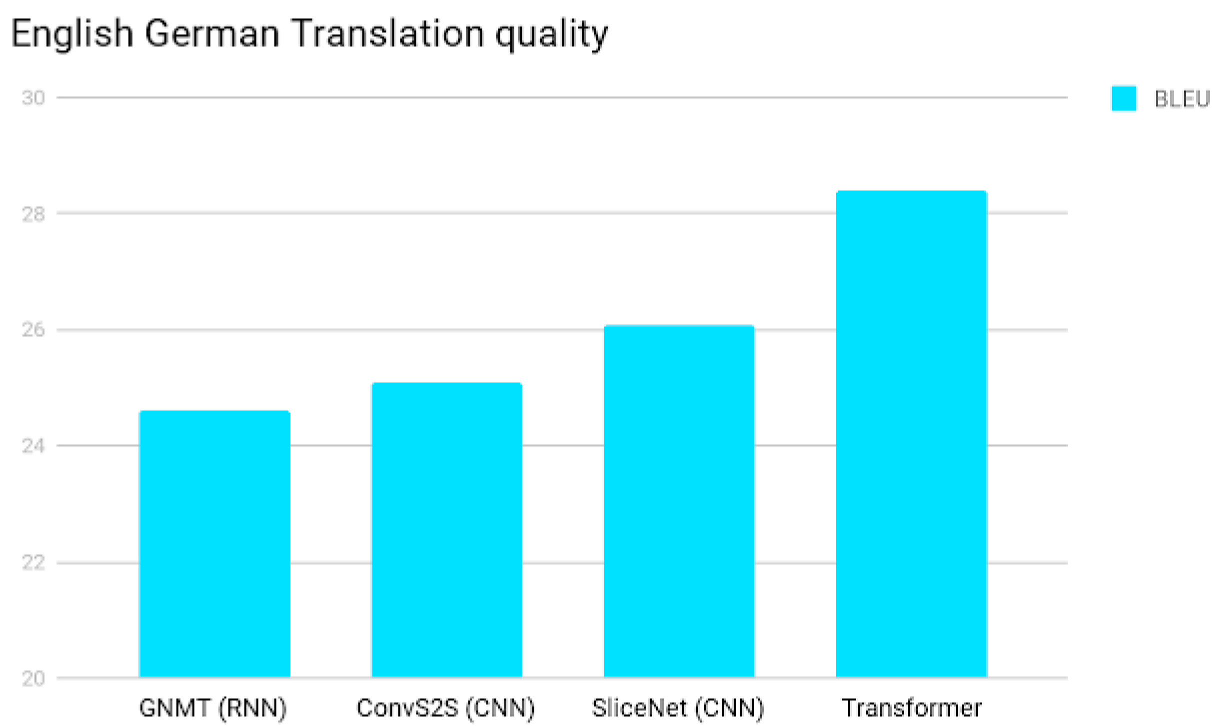
Dr Mehreen Alam

https://jalammar.github.io/illustrated-transformer/

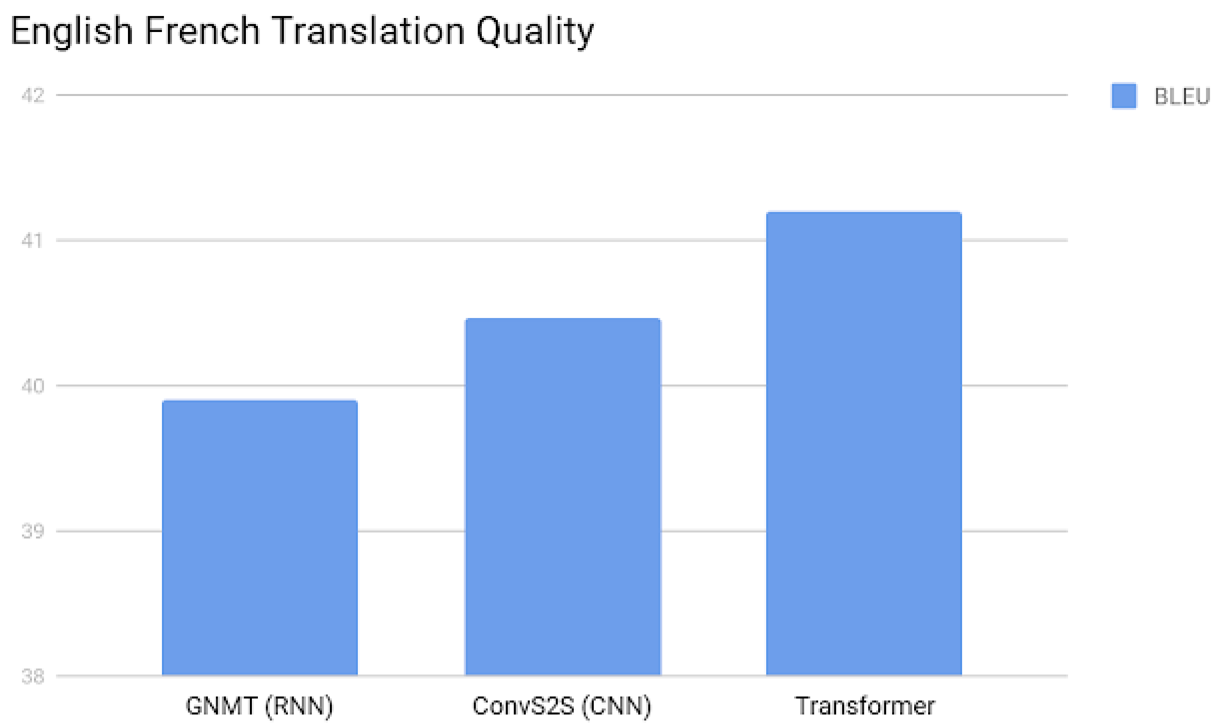
https://machinelearningmastery.com/a-gentle-introduction-to-positional-encoding-in-transformer-models-part-1/

Challenges of RNNs

- Long range dependencies
- Parallelization



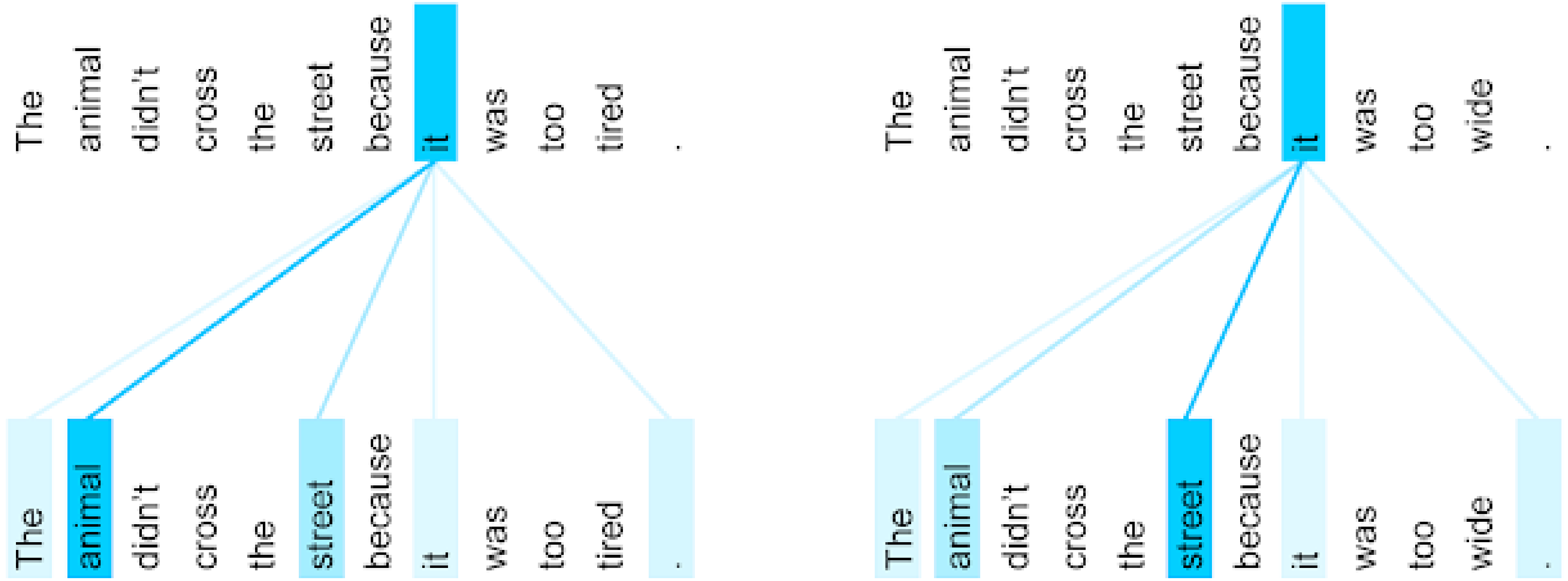
BLEU scores (higher is better) of single models on the standard WMT newstest2014 English to German translation benchmark.

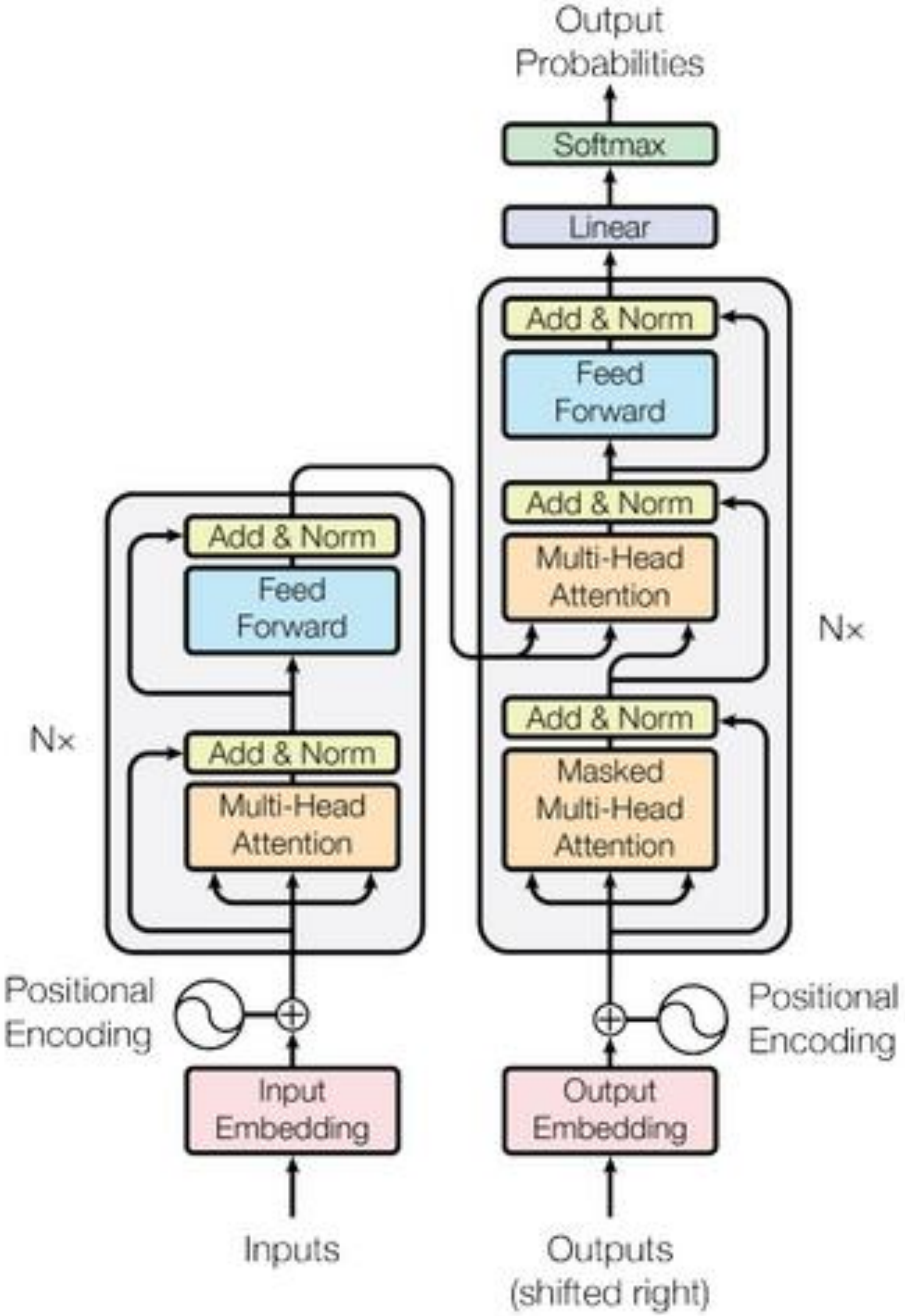


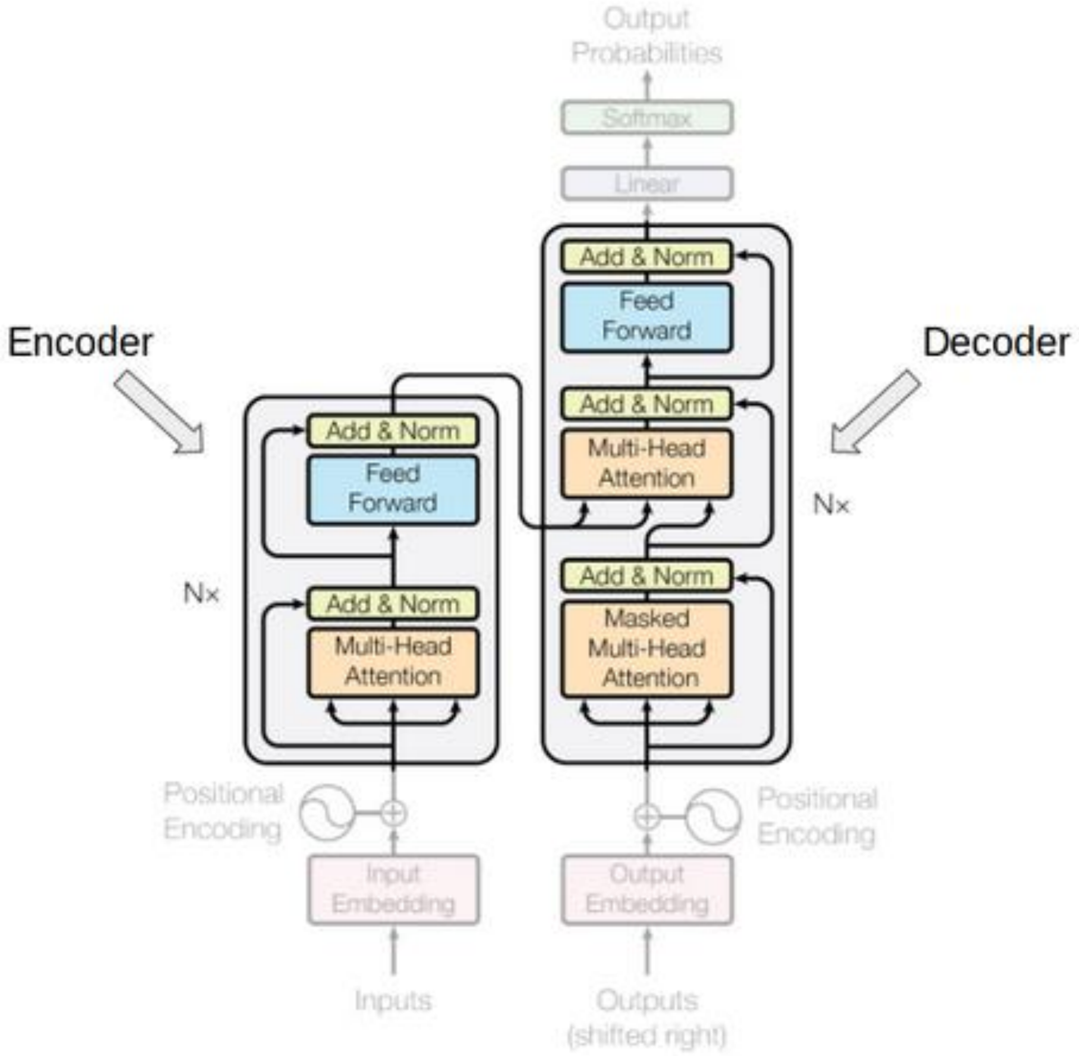
BLEU scores (higher is better) of single models on the standard WMT newstest2014 English to French translation benchmark.

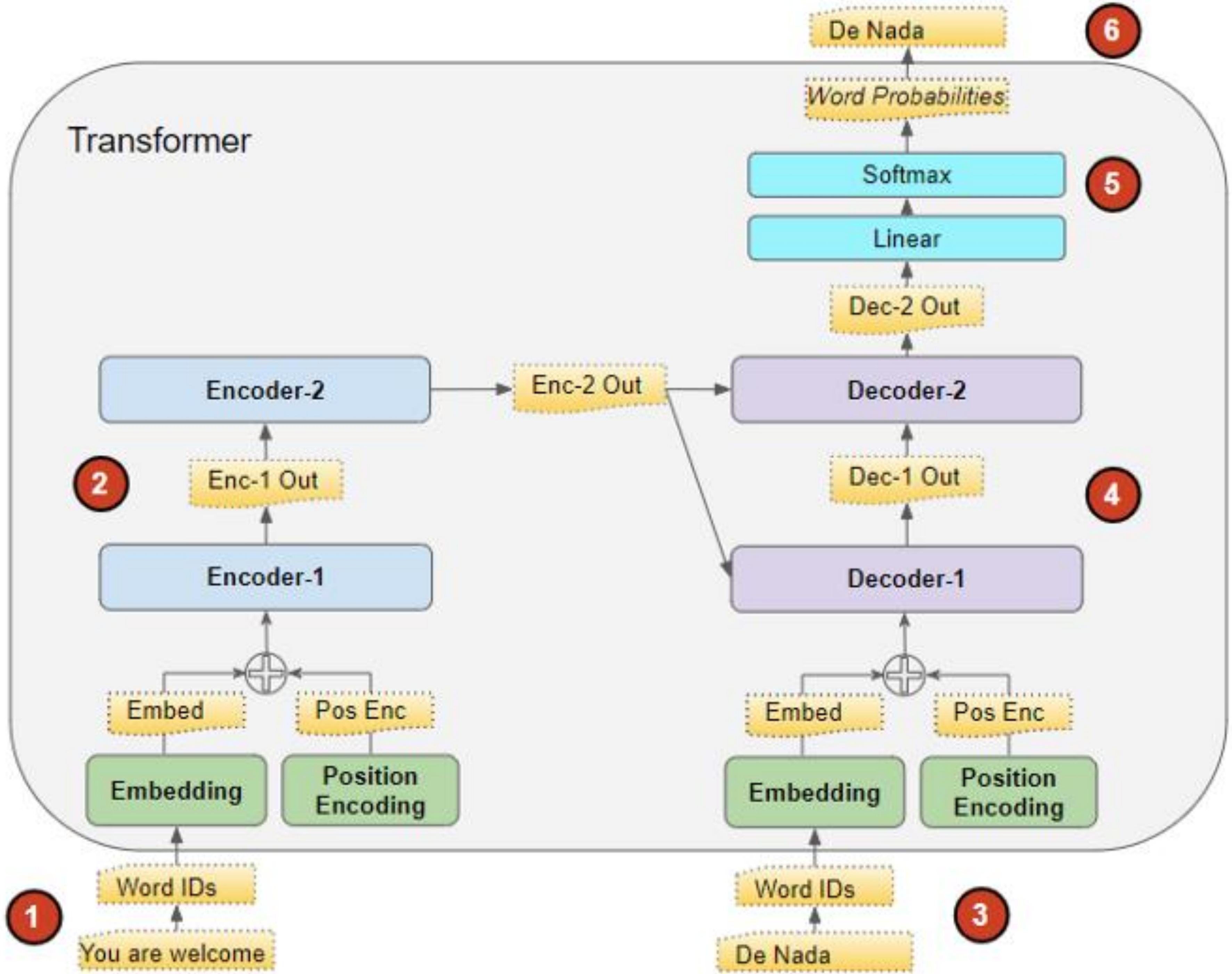
The animal didn't cross the street because it was too tired.
L'animal n'a pas traversé la rue parce qu'il était trop fatigué.

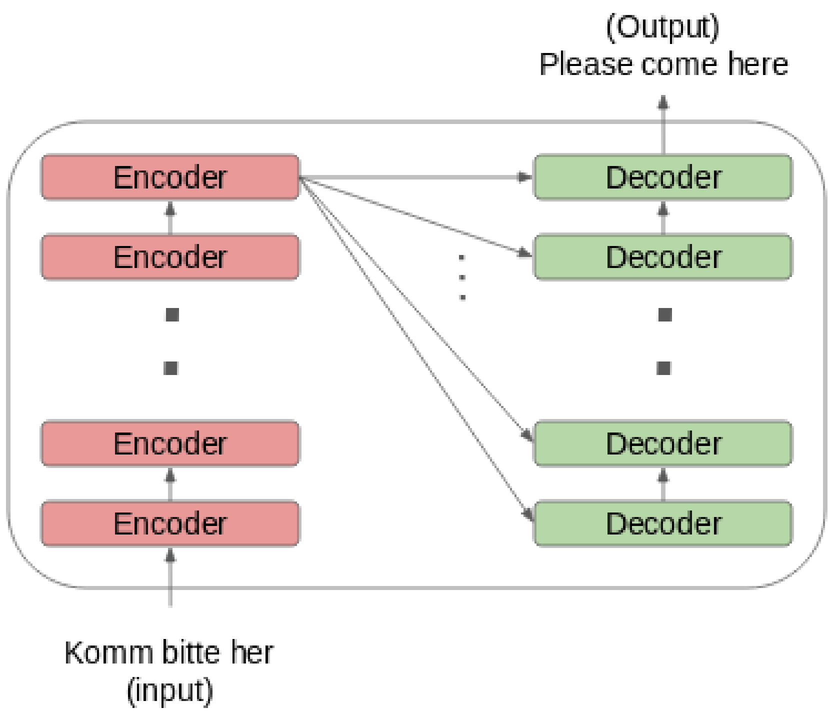
The animal didn't cross the street because it was too wide.
L'animal n'a pas traversé la rue parce qu'elle était trop large.

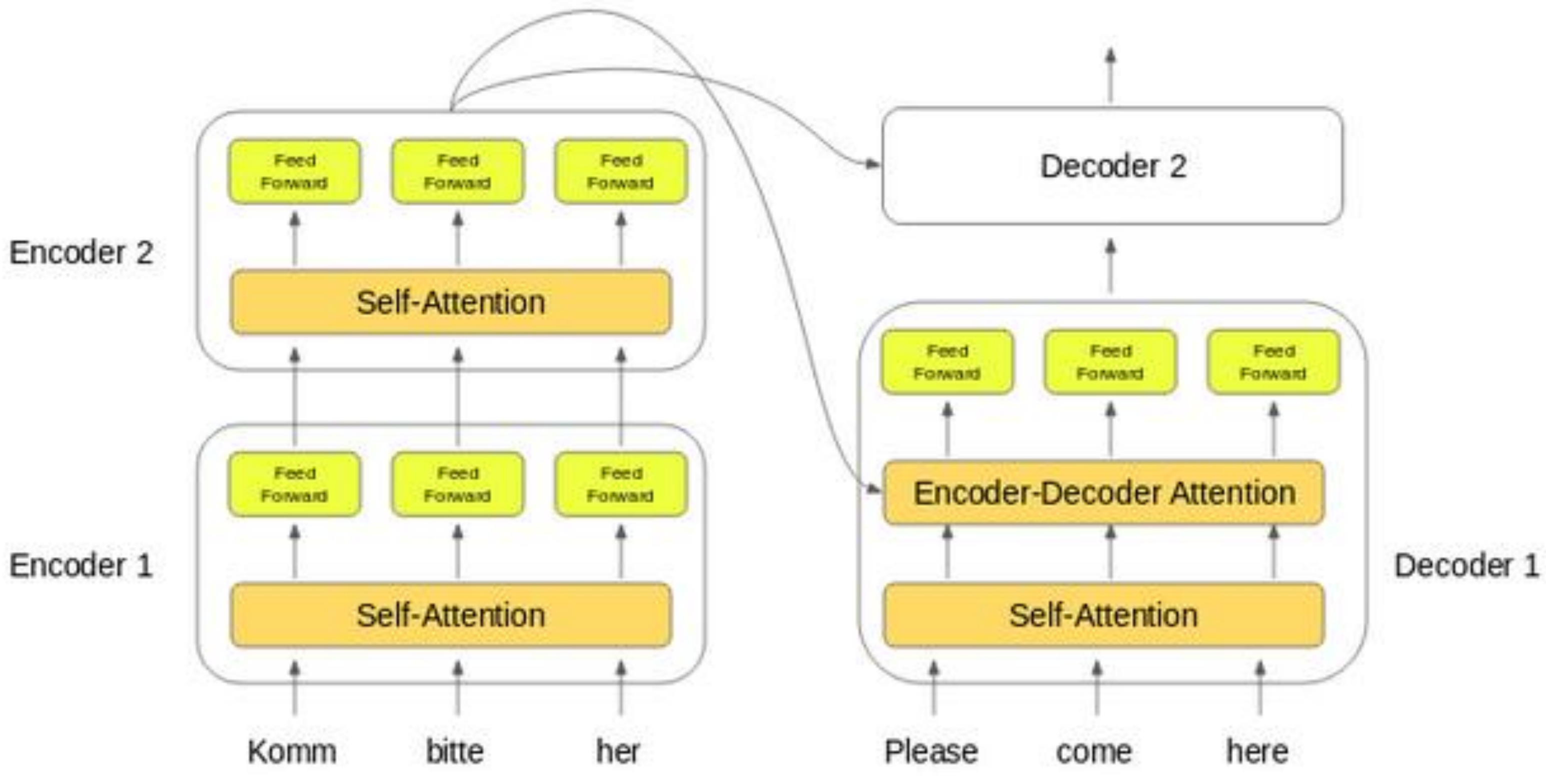



Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)


Encoder
Decoder
Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)


De Nada
6
Word Probabilities
Transformer
Softmax
5
Linear
Dec-2 Out
Encoder-2
Enc-2 Out
Decoder-2
2
Enc-1 Out
Dec-1 Out
4
Encoder-1
Decoder-1
Embed
Pos Enc
Embed
Pos Enc
Embedding
Position Encoding
Embedding
Position Encoding
Word IDs
Word IDs
1
You are welcome
3
De Nada


(Output)
Please come here
Encoder
Encoder
Encoder
Encoder
Decoder
Decoder
Decoder
Decoder
Komm bitte her
(input)


Encoder 2
Encoder 1
Feed Forward
Self-Attention
Decoder 2
Encoder-Decoder Attention
Decoder 1
Komm
bitte
her
Please
come
here

Self-Attention

- attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence.
- intra-attention

Calculating self-attention

1. create three vectors from each of the encoder's input vectors:
   1. Query Vector
   2. Key Vector
   3. Value Vector
2. calculate self-attention for every word in the input sequence
   1. Example for "Action gets results."

Word	q vector	k vector	v vector
Action	q_1	k_1	v_1
gets		k_2	v_2
results		k_3	v_3

Word	q vector	k vector	v vector	score
Action	q_1	k_1	v_1	$q_1 . k_1$
gets		k_2	v_2	$q_1 . k_2$
results		k_3	v_3	$q_1 . k_3$

Word	q vector	k vector	v vector	score	score / 8
Action	q_1	k_1	v_1	$q_1 . k_1$	$q_1 . k_1 / 8$
gets		k_2	v_2	$q_1 . k_2$	$q_1 . k_2 / 8$
results		k_3	v_3	$q_1 . k_3$	$q_1 . k_3 / 8$

Word	q vector	k vector	v vector	score	score / 8	Softmax
Action	q_1	k_1	v_1	$q_1 . k_1$	$q_1 . k_1 / 8$	x_{11}
gets		k_2	v_2	$q_1 . k_2$	$q_1 . k_2 / 8$	x_{12}
results		k_3	v_3	$q_1 . k_3$	$q_1 . k_3 / 8$	x_{13}

Word	q vector	k vector	v vector	score	score / 8	Softmax	Softmax * v	Sum
Action	q_1	k_1	v_1	$q_1 . k_1$	$q_1 . k_1 / 8$	x_{11}	$x_{11} * v_1$	z_1
gets		k_2	v_2	$q_1 . k_2$	$q_1 . k_2 / 8$	x_{12}	$x_{12} * v_2$	
results		k_3	v_3	$q_1 . k_3$	$q_1 . k_3 / 8$	x_{13}	$x_{13} * v_3$	

Word	q vector	k vector	v vector	score	score / 8	Softmax	Softmax * v	Sum#
Action		k_1	v_1	$q_2 . k_1$	$q_2 . k_1 / 8$	x_{21}	$x_{21} * v_1$	
gets	q_2	k_2	v_2	$q_2 . k_2$	$q_2 . k_2 / 8$	x_{22}	$x_{22} * v_2$	z_2
results		k_3	v_3	$q_2 . k_3$	$q_2 . k_3 / 8$	x_{23}	$x_{23} * v_3$	

Word	q vector	k vector	v vector	score	score / 8	Softmax	Softmax * v	Sum#
Action		k_1	v_1	$q_3 . k_1$	$q_3 . k_1 / 8$	x_{31}	$x_{31} * v_1$	
gets		k_2	v_2	$q_3 . k_2$	$q_3 . k_2 / 8$	x_{32}	$x_{32} * v_2$	
results	q_3	k_3	v_3	$q_3 . k_3$	$q_3 . k_3 / 8$	x_{33}	$x_{33} * v_3$	z_3

Recap of Attention from Seq2Seq

Attention: in equations

- We have encoder hidden states $h_1, \ldots, h_N \in \mathbb{R}^h$
- On timestep *t*, we have decoder hidden state $s_t \in \mathbb{R}^h$
- We get the attention scores $\boldsymbol{e}^t$ for this step:

$$\boldsymbol{e}^t = [\boldsymbol{s}_t^T \boldsymbol{h}_1, \ldots, \boldsymbol{s}_t^T \boldsymbol{h}_N] \in \mathbb{R}^N$$

- We take softmax to get the attention distribution α^t for this step (this is a probability distribution and sums to 1)

$$\alpha^t = \mathrm{softmax}(\boldsymbol{e}^t) \in \mathbb{R}^N$$

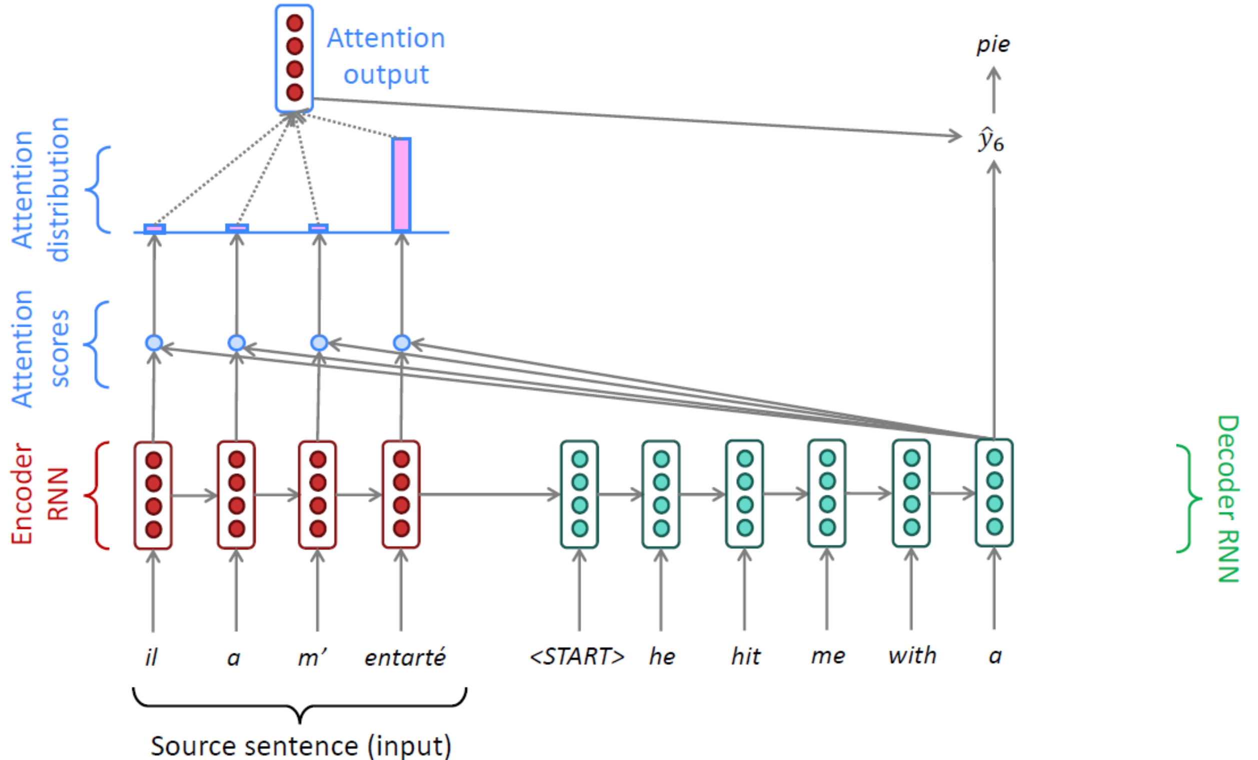
- We use α^t to take a weighted sum of the encoder hidden states to get the attention output $\boldsymbol{a}_t$

$$\boldsymbol{a}_t = \sum_{i=1}^{N} \alpha_i^t \boldsymbol{h}_i \in \mathbb{R}^h$$

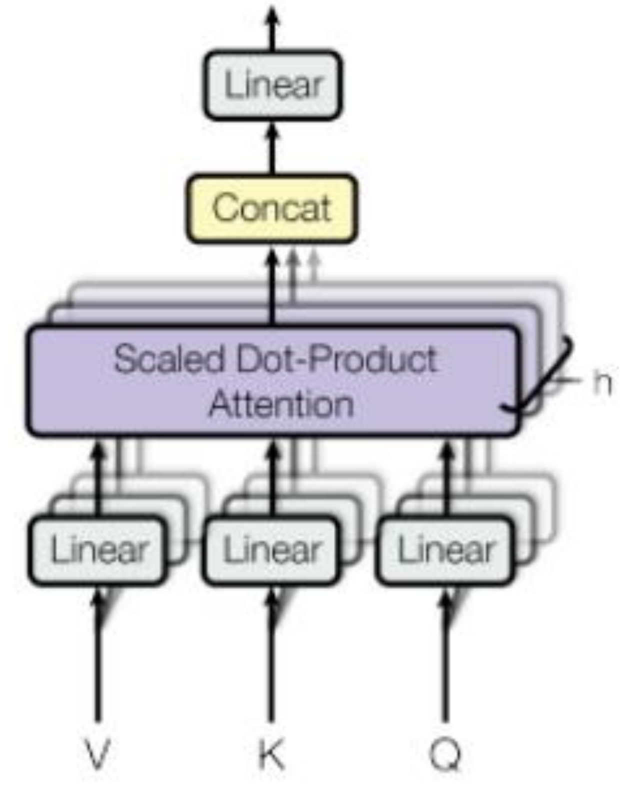
- Finally we concatenate the attention output $\boldsymbol{a}_t$ with the decoder hidden state s_t and proceed as in the non-attention seq2seq model

$$[\boldsymbol{a}_t; \boldsymbol{s}_t] \in \mathbb{R}^{2h}$$

ence-to-sequence with attention

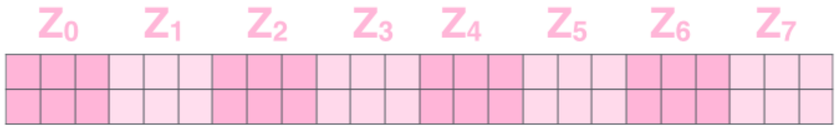


Multi-headed Attention



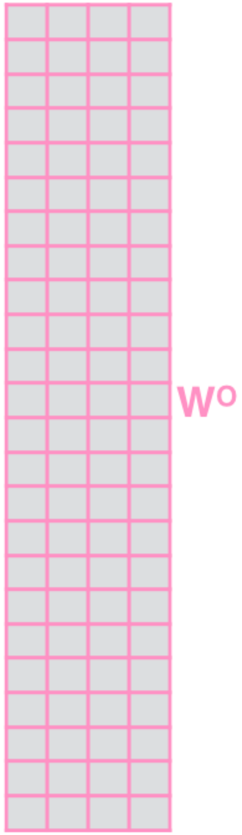
Multi-Head Attention

1) Concatenate all the attention heads



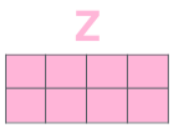
2) Multiply with a weight matrix W^O that was trained jointly with the model

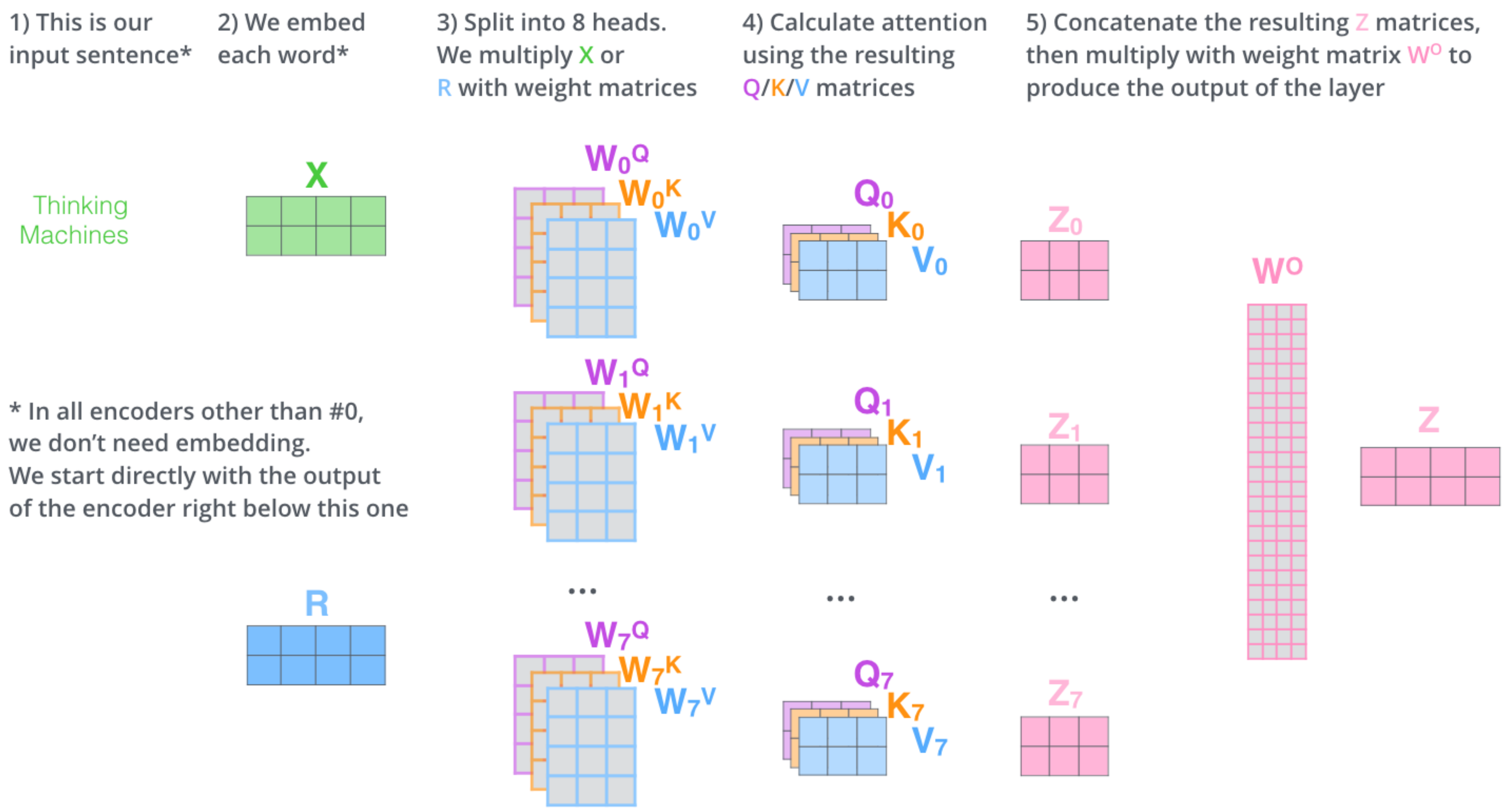
x

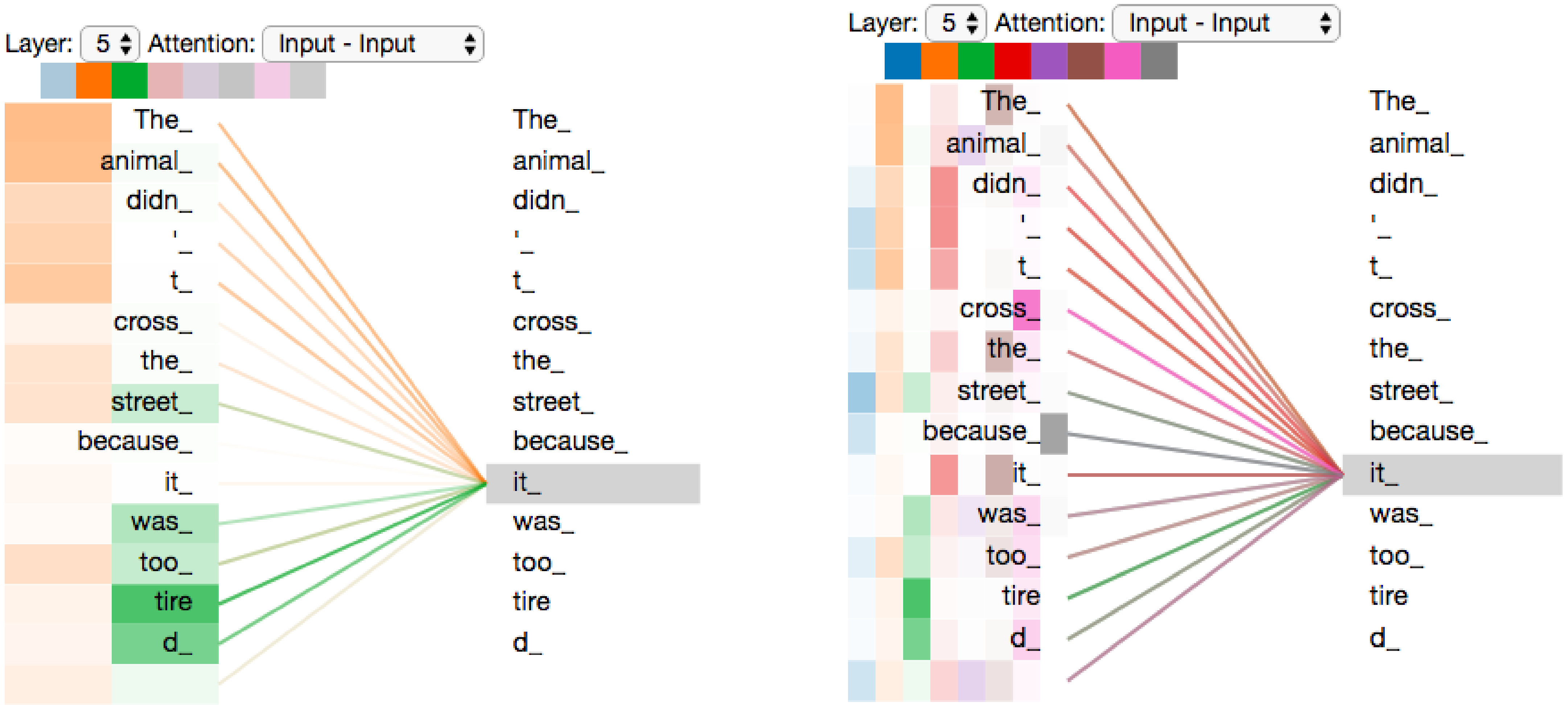


3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN

=

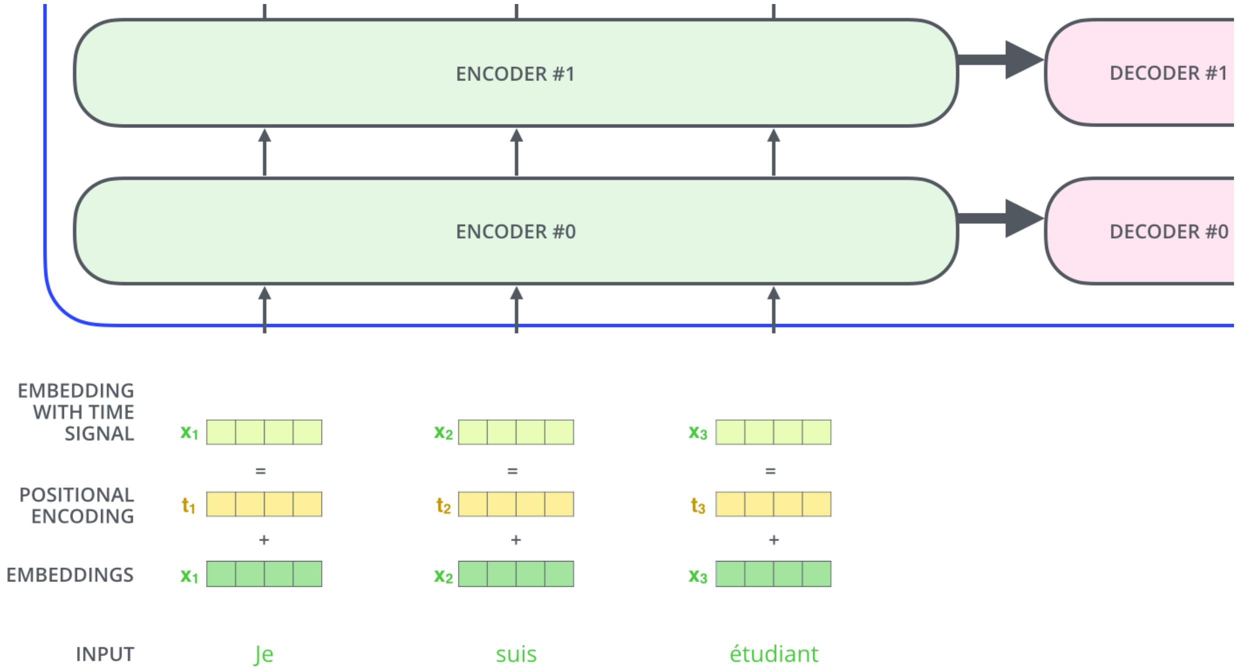


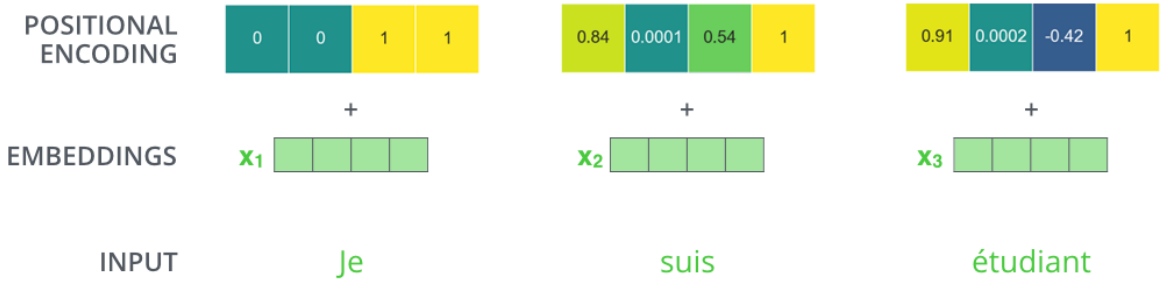

1) This is our input sentence*
2) We embed each word*
3) Split into 8 heads. We multiply X or R with weight matrices
4) Calculate attention using the resulting Q/K/V matrices
5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer
Thinking Machines
X
W0Q
W0K
W0V
Q0
K0
V0
Z0
WO
* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one
W1Q
W1K
W1V
Q1
K1
V1
Z1
Z
R
...
W7Q
W7K
W7V
Q7
K7
V7
Z7

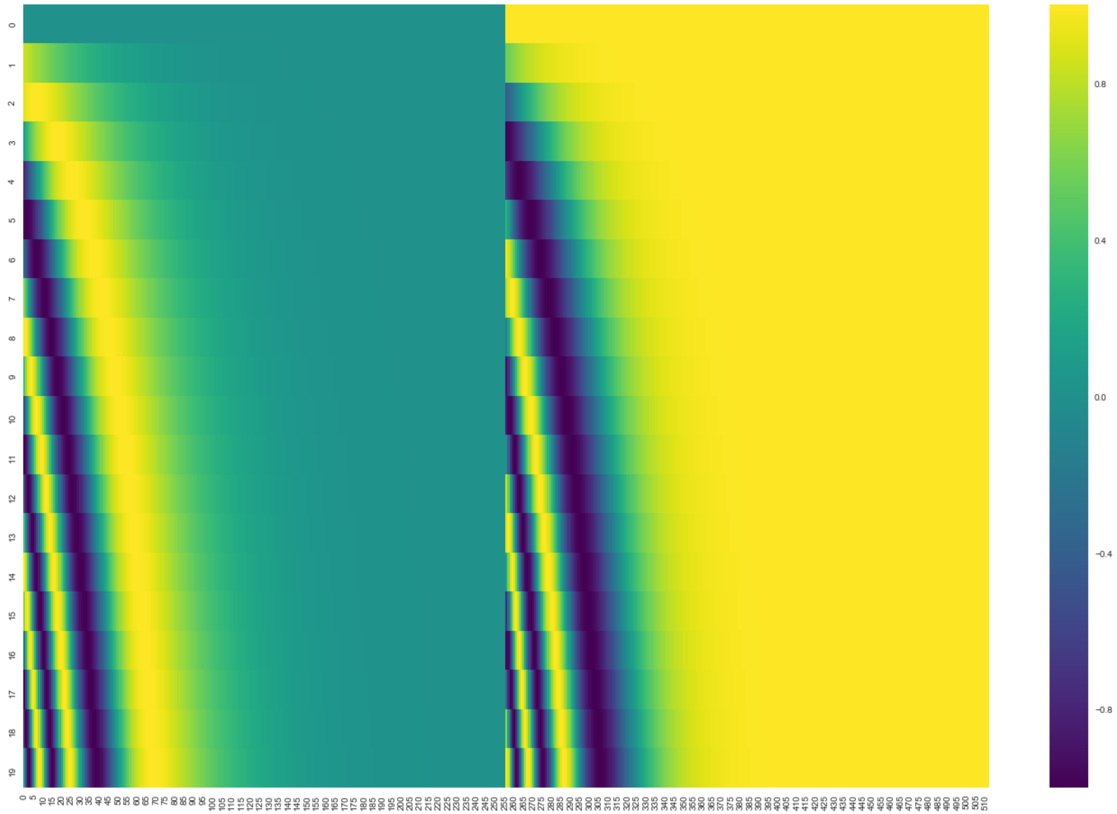


Layer: 5 Attention: Input - Input
The_
animal_
didn_
'_
t_
cross_
the_
street_
because_
it_
was_
too_
tire
d_
Layer: 5 Attention: Input - Input
The_
animal_
didn_
'_
t_
cross_
the_
street_
because_
it_
was_
too_
tire
d_

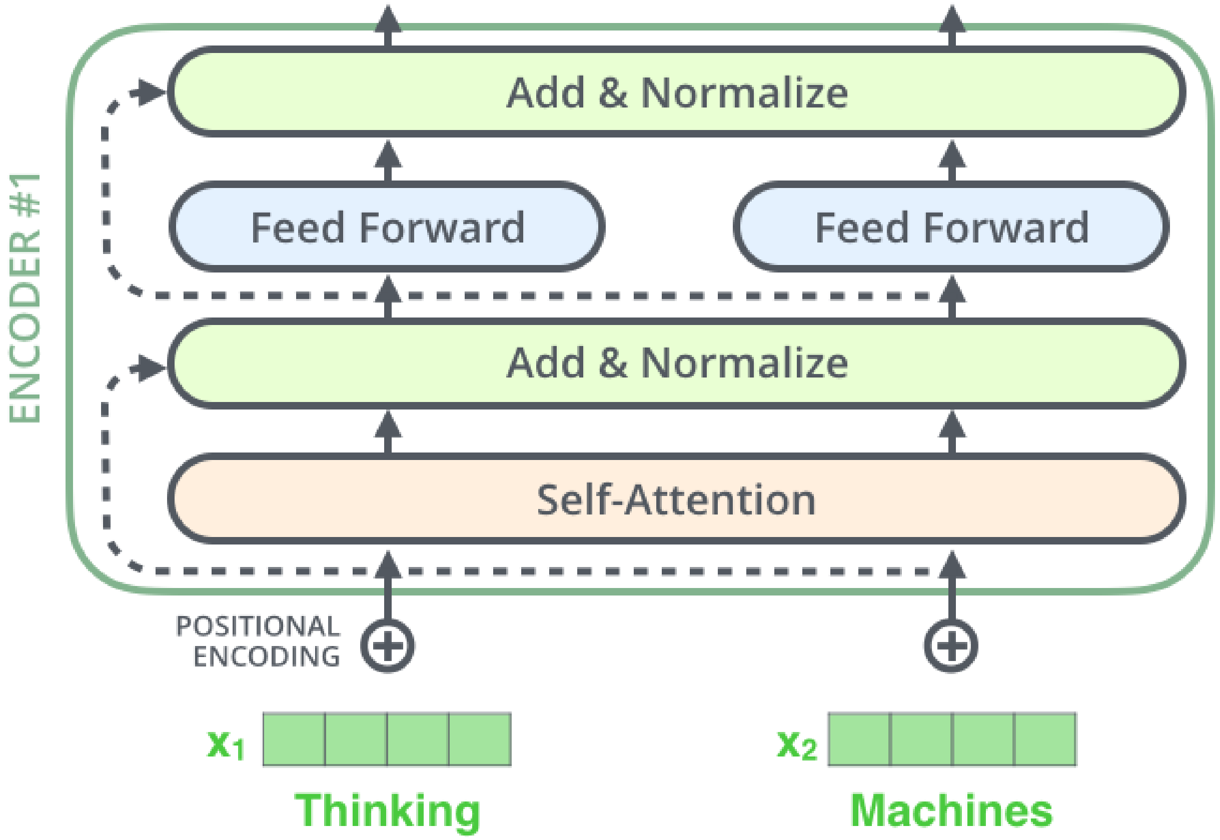
Positional Encoding and Embedding

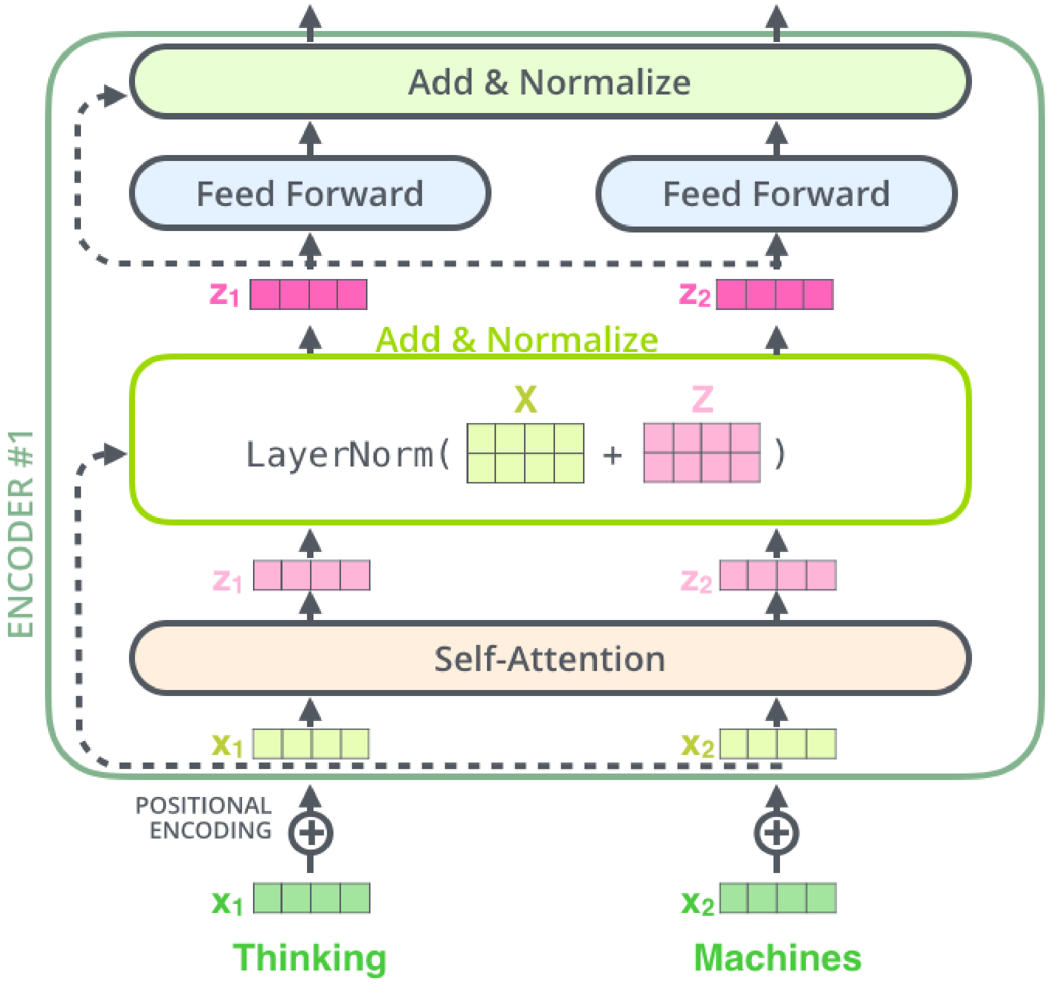




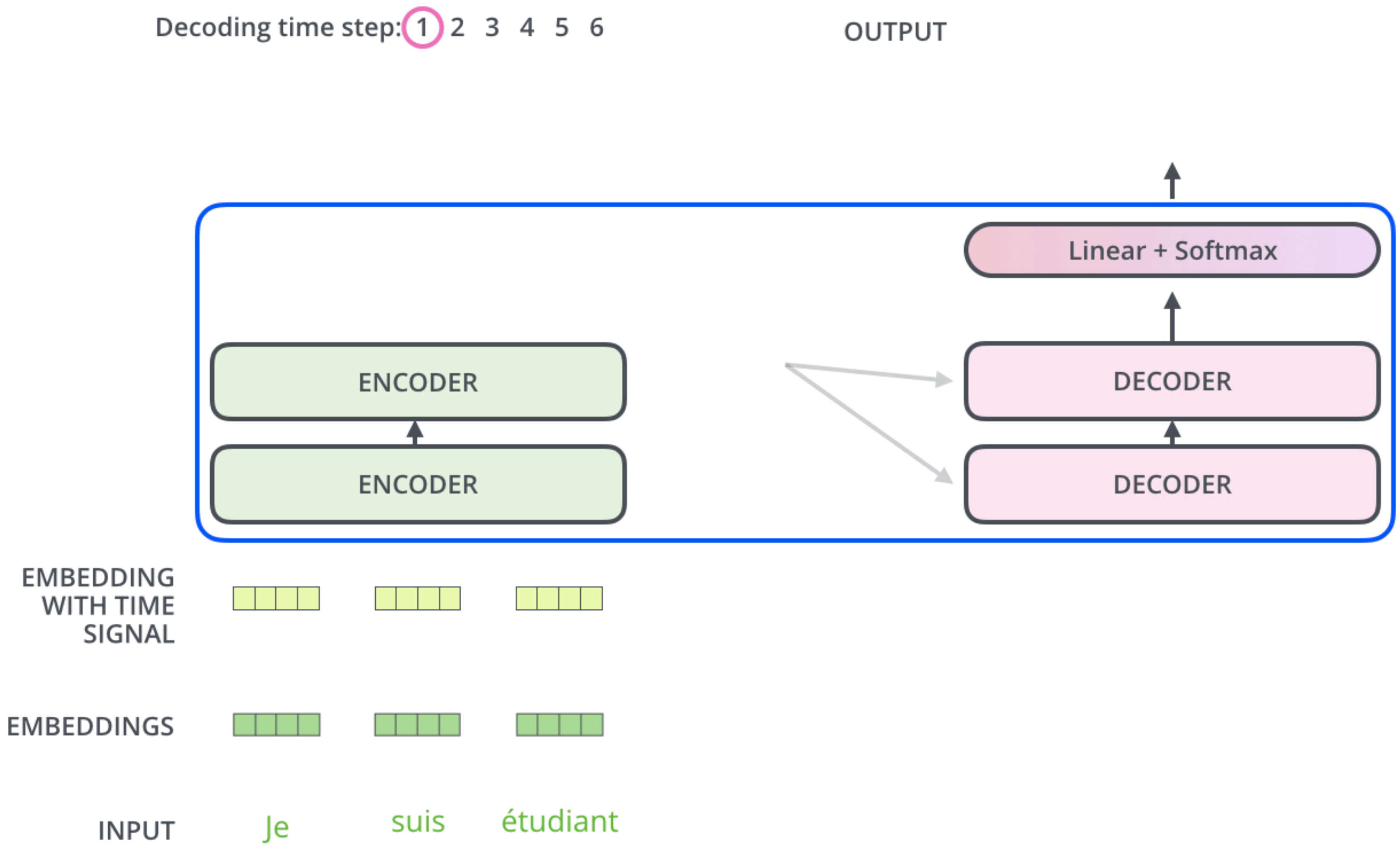


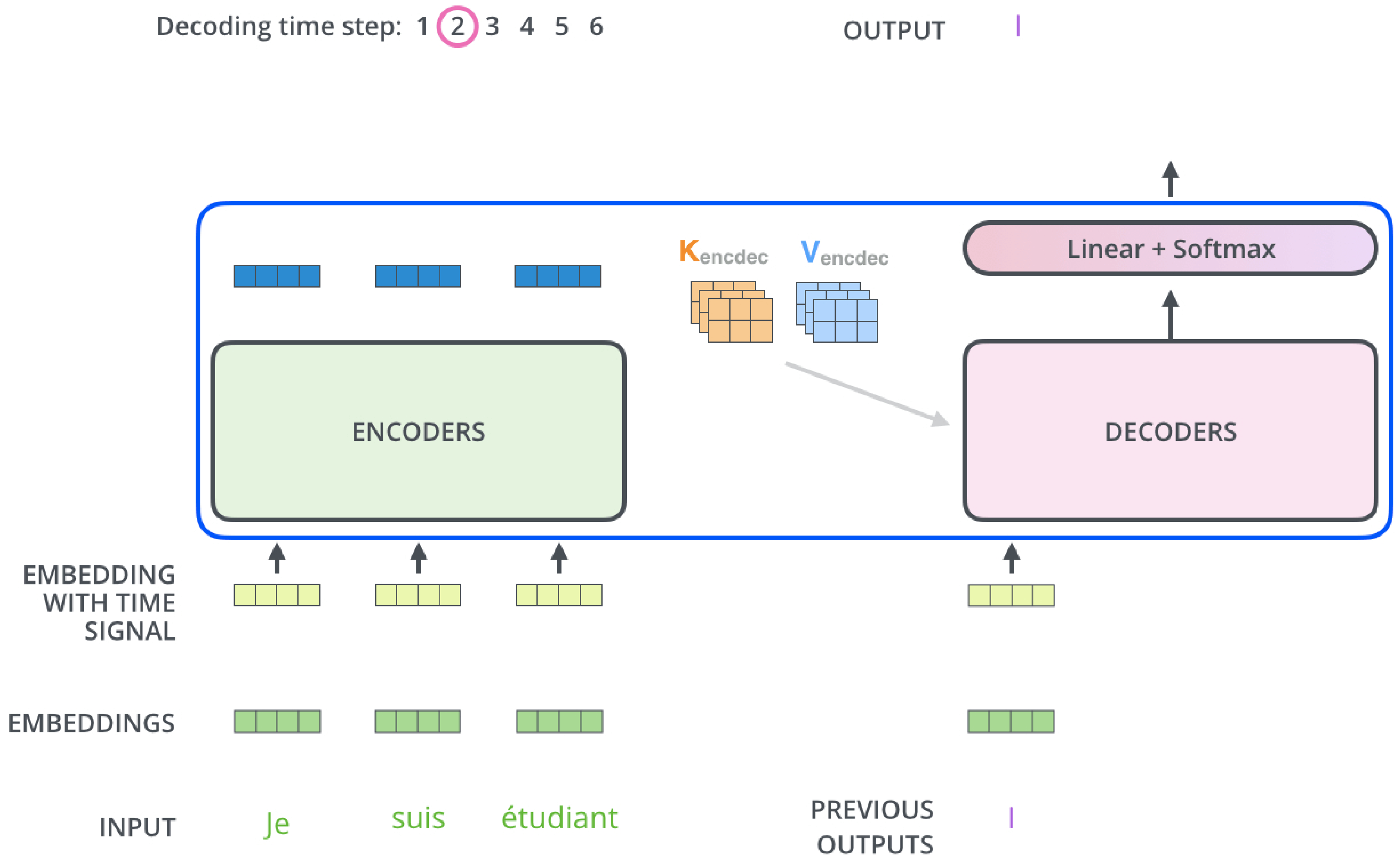
Residuals


Add & Normalize
Feed Forward
Feed Forward
Add & Normalize
Self-Attention
ENCODER #1
POSITIONAL ENCODING
x_1
Thinking
x_2
Machines

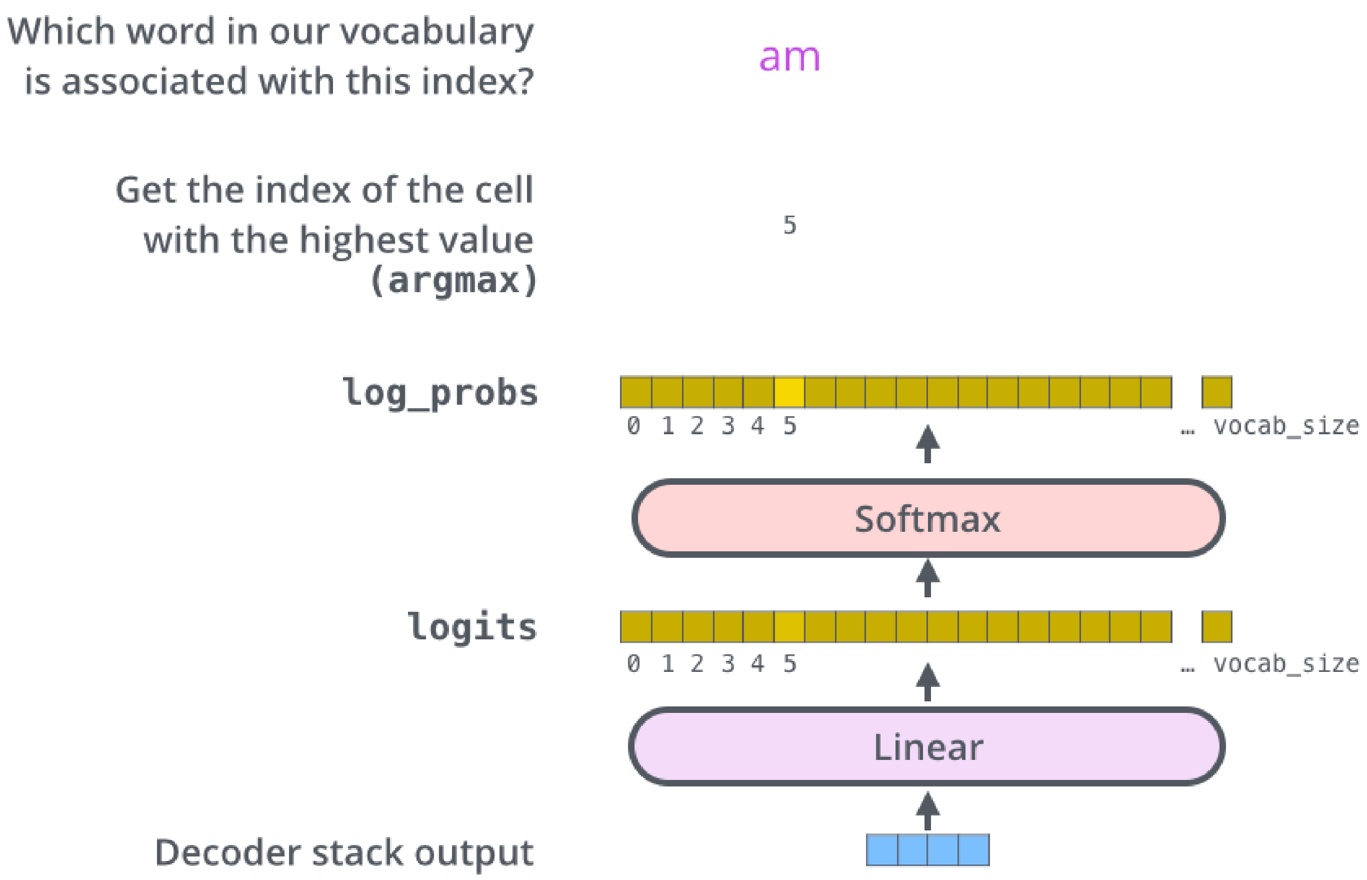

Add & Normalize
Feed Forward
Feed Forward
z_1
z_2
Add & Normalize
X
Z
LayerNorm(+)
z_1
z_2
Self-Attention
x_1
x_2
ENCODER #1
POSITIONAL ENCODING
x_1
x_2
Thinking
Machines

Decoder


Decoding time step: 1 2 3 4 5 6
OUTPUT
Linear + Softmax
ENCODER
ENCODER
DECODER
DECODER
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant



Decoding time step: 1 2 3 4 5 6
OUTPUT
I
Linear + Softmax
Kencdec
Vencdec
ENCODERS
DECODERS
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant
PREVIOUS OUTPUTS
I



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
0 1 2 3 4 5 … vocab_size
Softmax
logits
0 1 2 3 4 5 … vocab_size
Linear
Decoder stack output

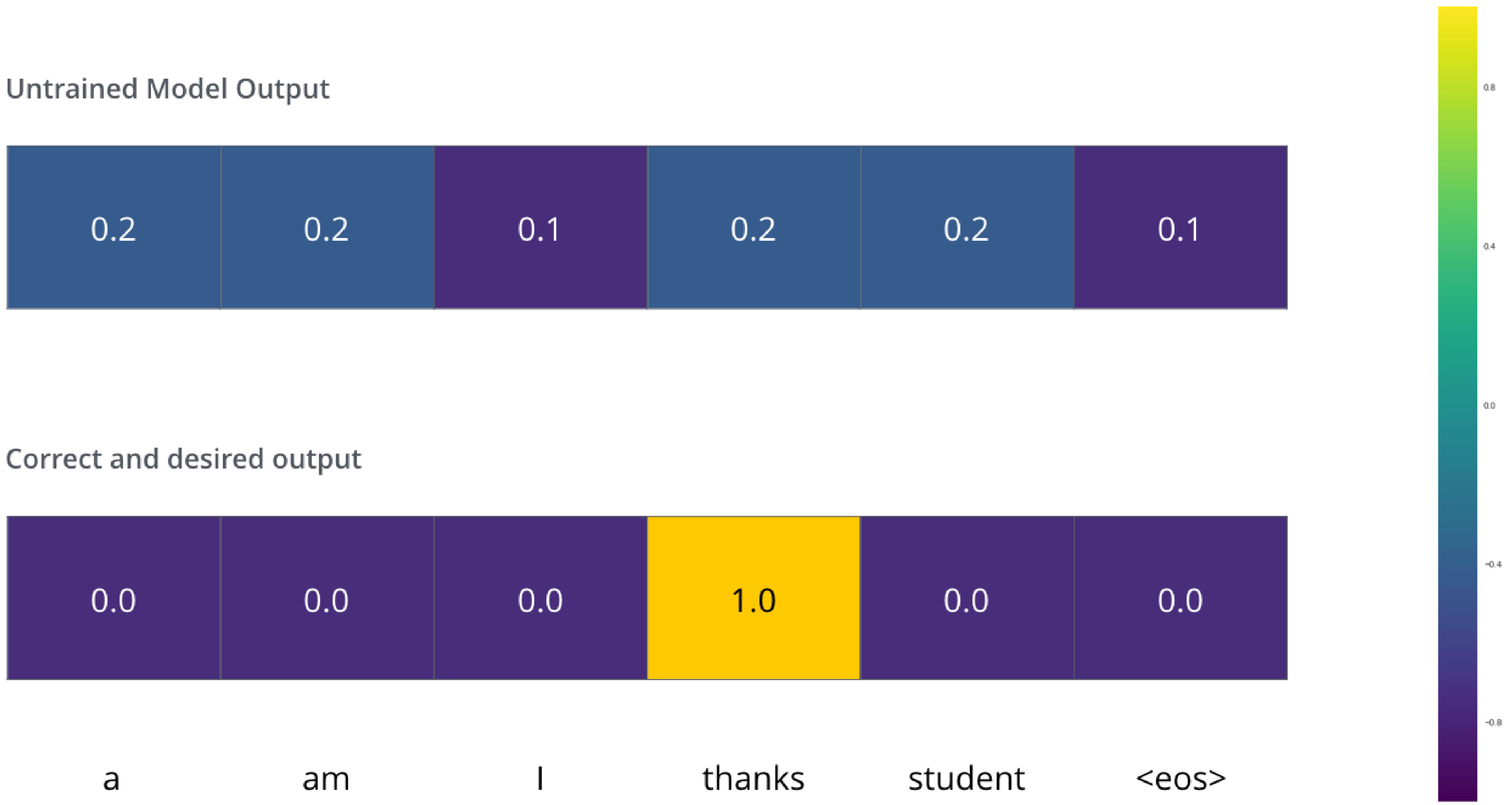
Training Details

Output Vocabulary

WORD	a	am	I	thanks	student	<eos>
INDEX	0	1	2	3	4	5

One-hot encoding of the word “am”

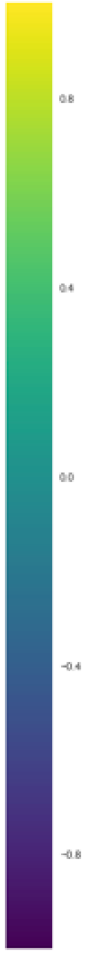
0.0	1.0	0.0	0.0	0.0	0.0



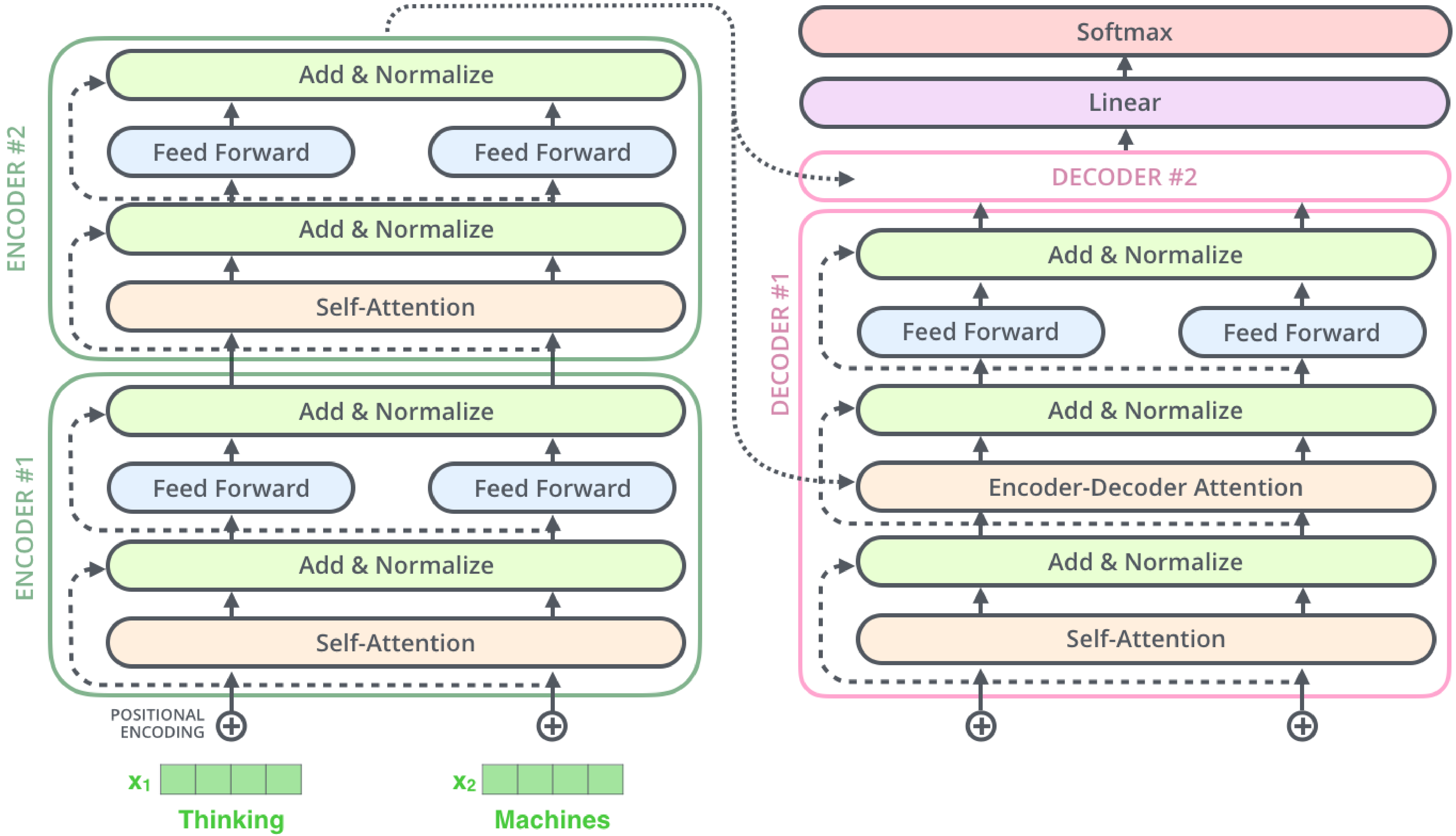
Untrained Model Output
0.2
0.2
0.1
0.2
0.2
0.1
Correct and desired output
0.0
0.0
0.0
1.0
0.0
0.0
a
am
I
thanks
student
<eos>
0.8
0.4
0.0
-0.4
-0.8

Trained Model Outputs

Output Vocabulary:	a	am	I	thanks	student	<eos>
position #1	0.01	0.02	0.93	0.01	0.03	0.01
position #2	0.01	0.8	0.1	0.05	0.01	0.03
position #3	0.99	0.001	0.001	0.001	0.002	0.001
position #4	0.001	0.002	0.001	0.02	0.94	0.01
position #5	0.01	0.01	0.001	0.001	0.001	0.98



a am I thanks student <eos>


ENCODER #2
Add & Normalize
Feed Forward
Feed Forward
Add & Normalize
Self-Attention
ENCODER #1
Add & Normalize
Feed Forward
Feed Forward
Add & Normalize
Self-Attention
POSITIONAL
ENCODING
x_1
Thinking
x_2
Machines
Softmax
Linear
DECODER #2
DECODER #1
Add & Normalize
Feed Forward
Feed Forward
Add & Normalize
Encoder-Decoder Attention
Add & Normalize
Self-Attention

Limitations of Transformers

- Attention can only deal with fixed-length text strings:
  - The text has to be split into a certain number of segments before being fed into the system as input.
- This chunking of text causes context fragmentation:
  - E.g., if a sentence is split from the middle, then a significant amount of context is lost.
  - In other words, the text is split without respecting the sentence or any other semantic boundary

Useful Links

- Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems. 2017.
- https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html
- https://www.analyticsvidhya.com/blog/2019/06/understanding-transformers-nlp-state-of-the-art-models/
- https://jalammar.github.io/illustrated-transformer/
- Sample Code Links:
  - https://github.com/tensorflow/tensor2tensor/blob/master/README.md#walkthrough
  - https://colab.research.google.com/github/tensorflow/tensor2tensor/blob/master/tensor2tensor/notebooks/hello_t2t.ipynb#scrollTo=s19ucTii_wYb