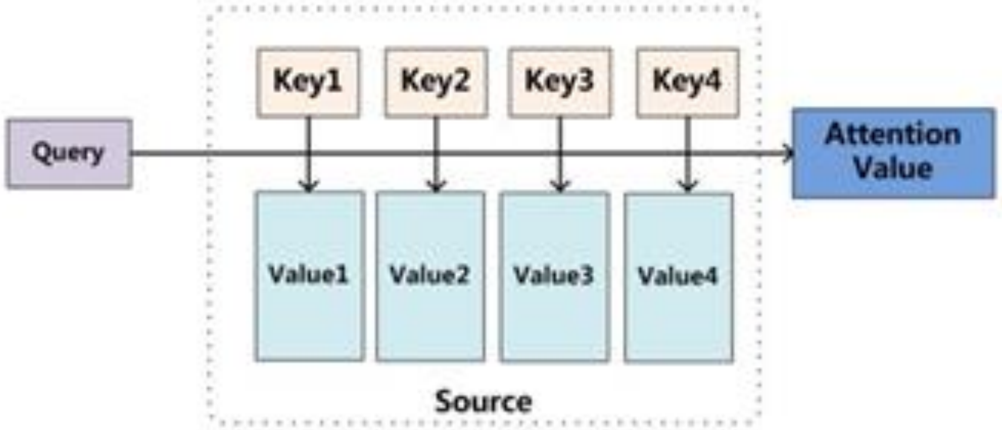
Attention is all you need

Steven Tang

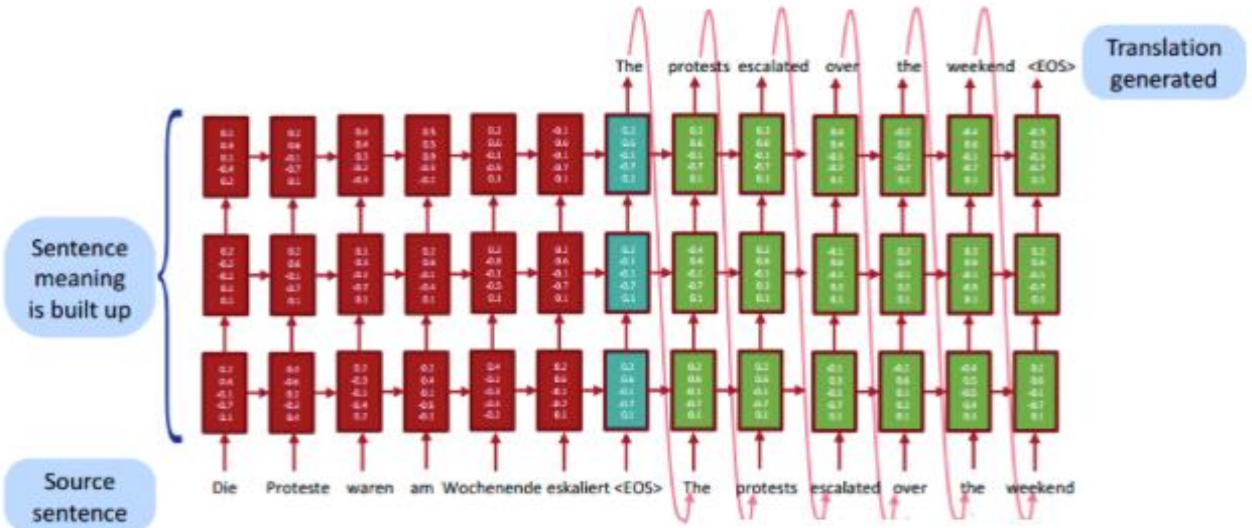
注意力机制回顾

- An attention function can be described as **mapping a query and a set of key-value pairs** to an output.

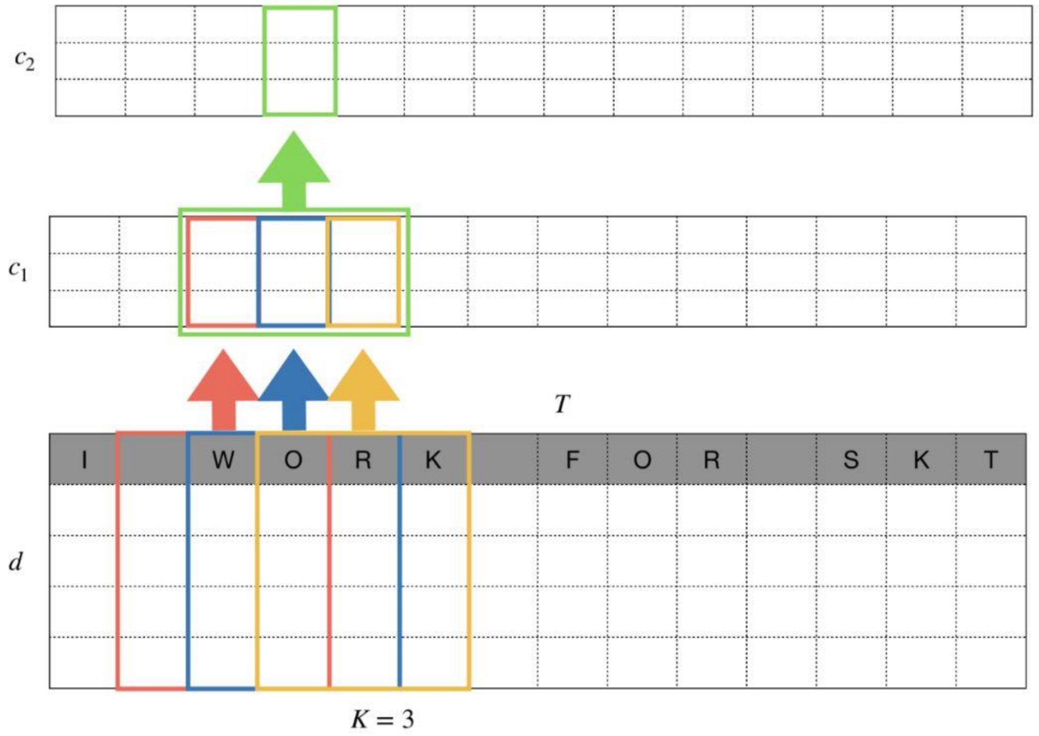


$$\text{Attention}(\text{Query}, \text{Source}) = \sum_{i=1}^{L_x} Similarity(Query, Key_i) * Value_i$$

CNN和RNN的问题

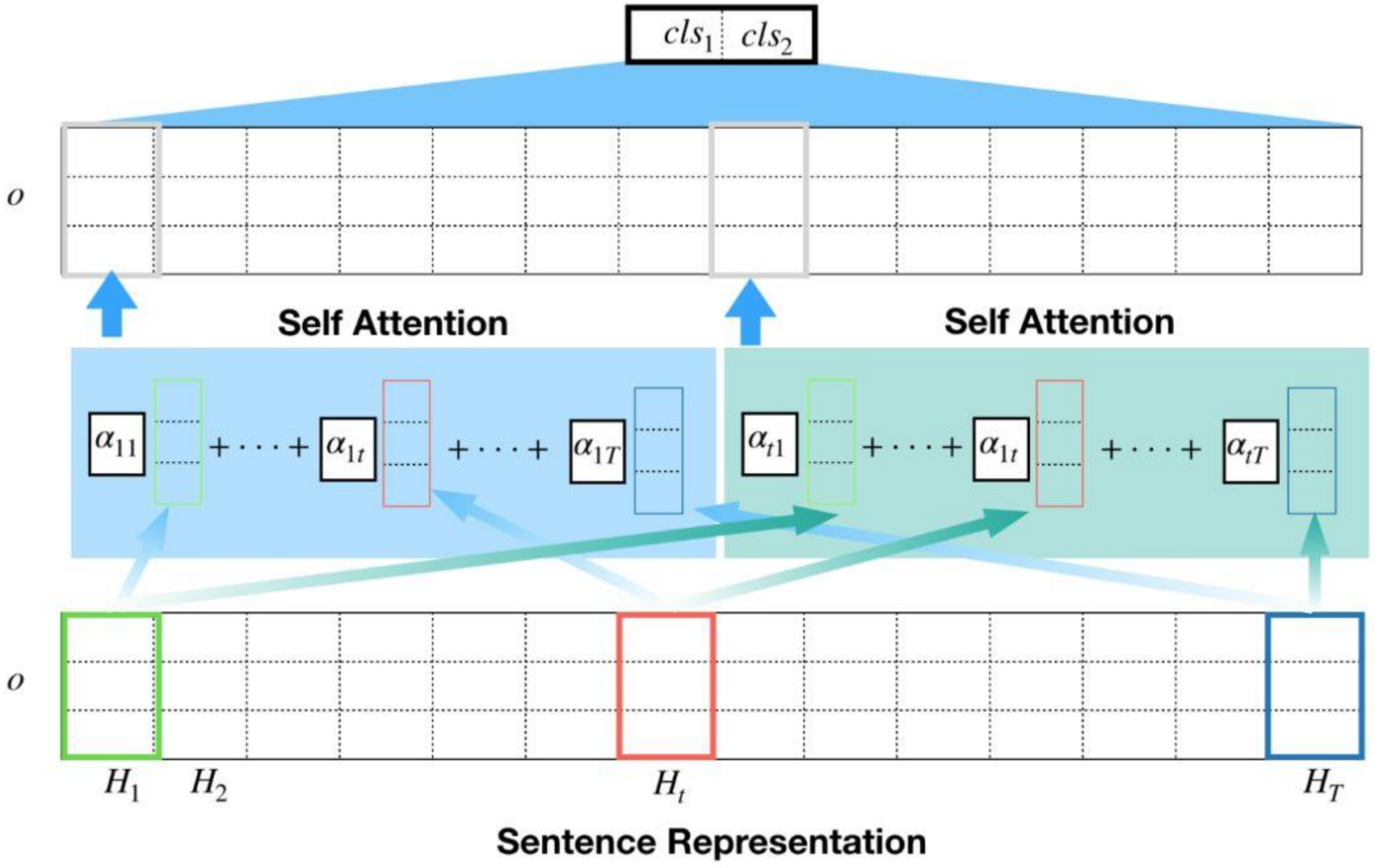


Translation generated
The
protests
escalated
over
the
weekend
<EOS>
Sentence meaning is built up
Source sentence
Die
Proteste
waren
am
Wochenende
eskaliert
<EOS>
The
protests
escalated
over
the
weekend



c_2
c_1
T
I
W
O
R
K
F
O
R
S
K
T
d
$K = 3$

自注意力机制



cls_1 cls_2
o
Self Attention
Self Attention
α_{11} + · · · + α_{1t} + · · · + α_{1T}
α_{t1} + · · · + α_{1t} + · · · + α_{tT}
o
H_1 H_2 H_t H_T
Sentence Representation

自注意力横跨出世

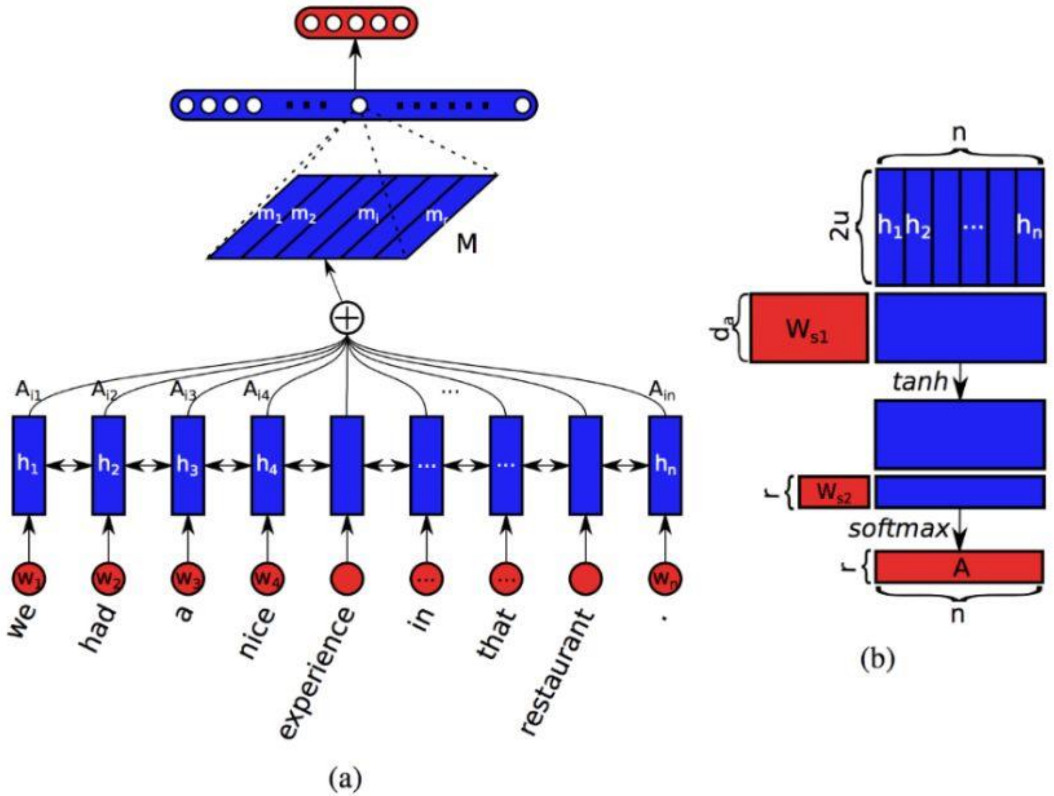
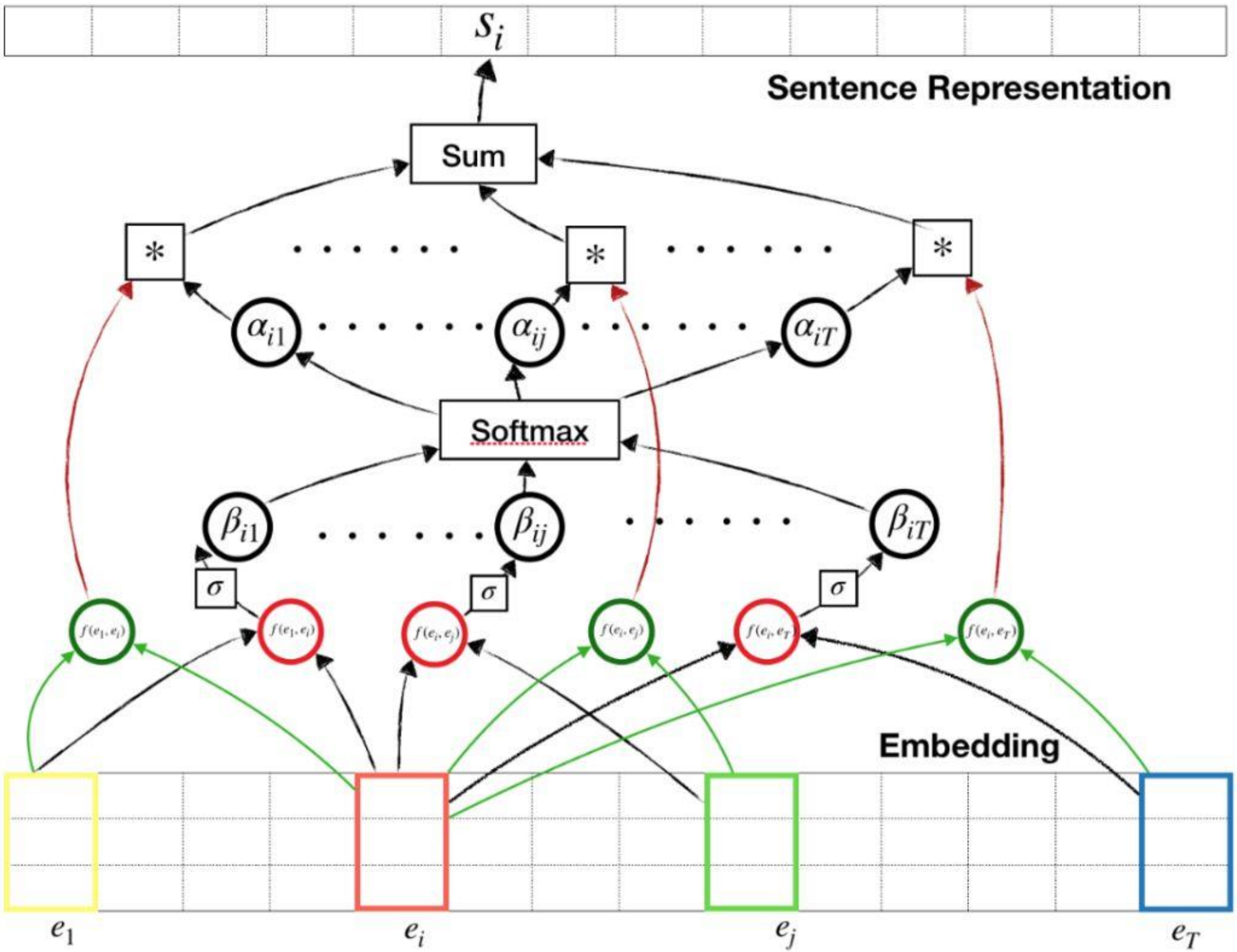
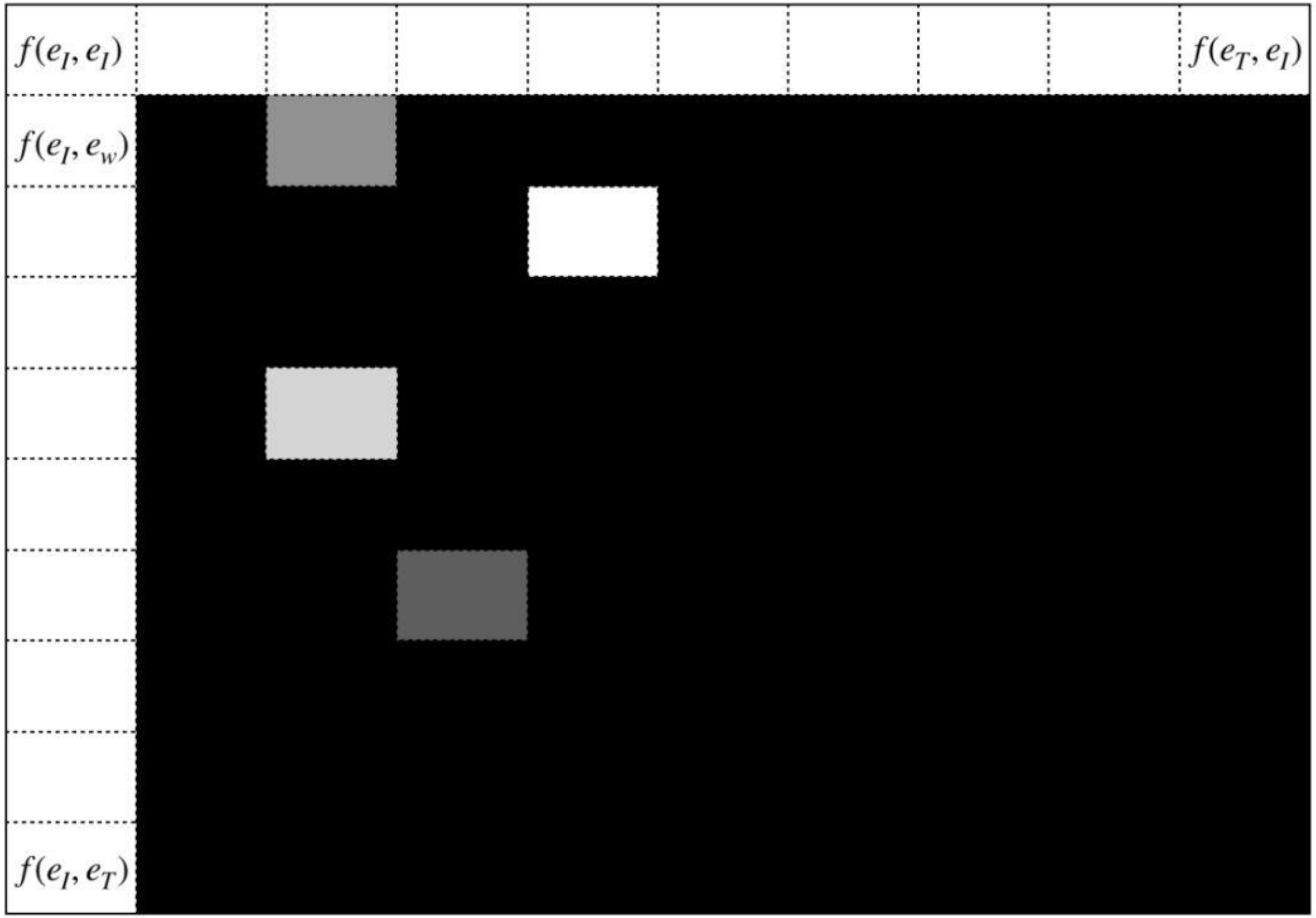


Figure 1: A sample model structure showing the sentence embedding model combined with a fully connected and softmax layer for sentiment analysis (a). The sentence embedding M is computed as multiple weighted sums of hidden states from a bidirectional LSTM $(\mathbf{h_1}, ..., \mathbf{h_n})$, where the summation weights $(A_{i1}, ..., A_{in})$ are computed in a way illustrated in (b). Blue colored shapes stand for hidden representations, and red colored shapes stand for weights, annotations, or input/output.



s_i
Sentence Representation
Sum
*
α_{i1}
α_{ij}
α_{iT}
Softmax
β_{i1}
β_{ij}
β_{iT}
σ
Embedding
e_1
e_i
e_j
e_T

自注意力机制



Self-attention（自注意力机制）

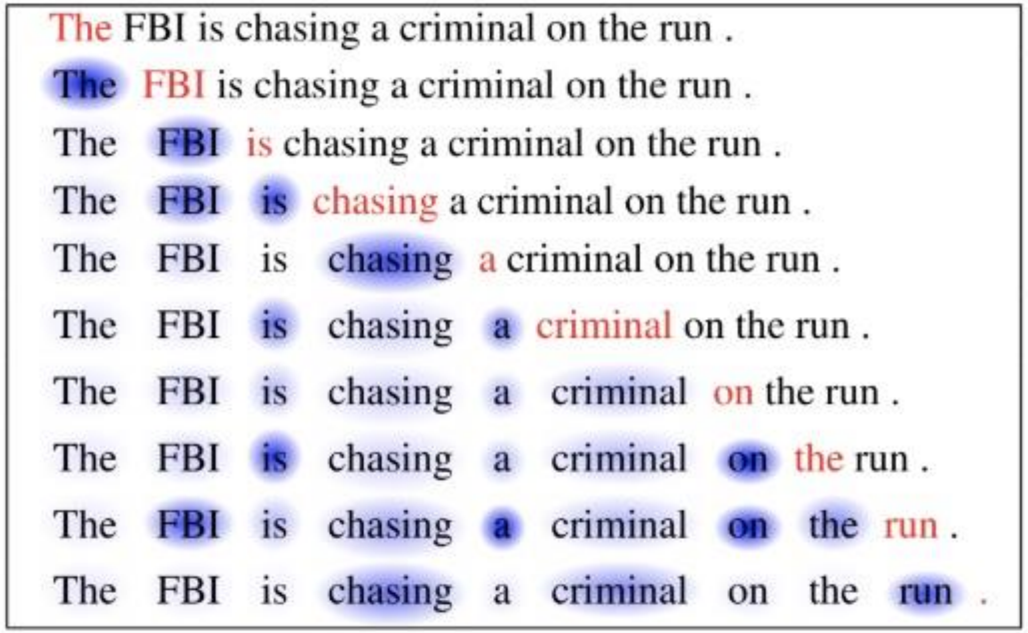
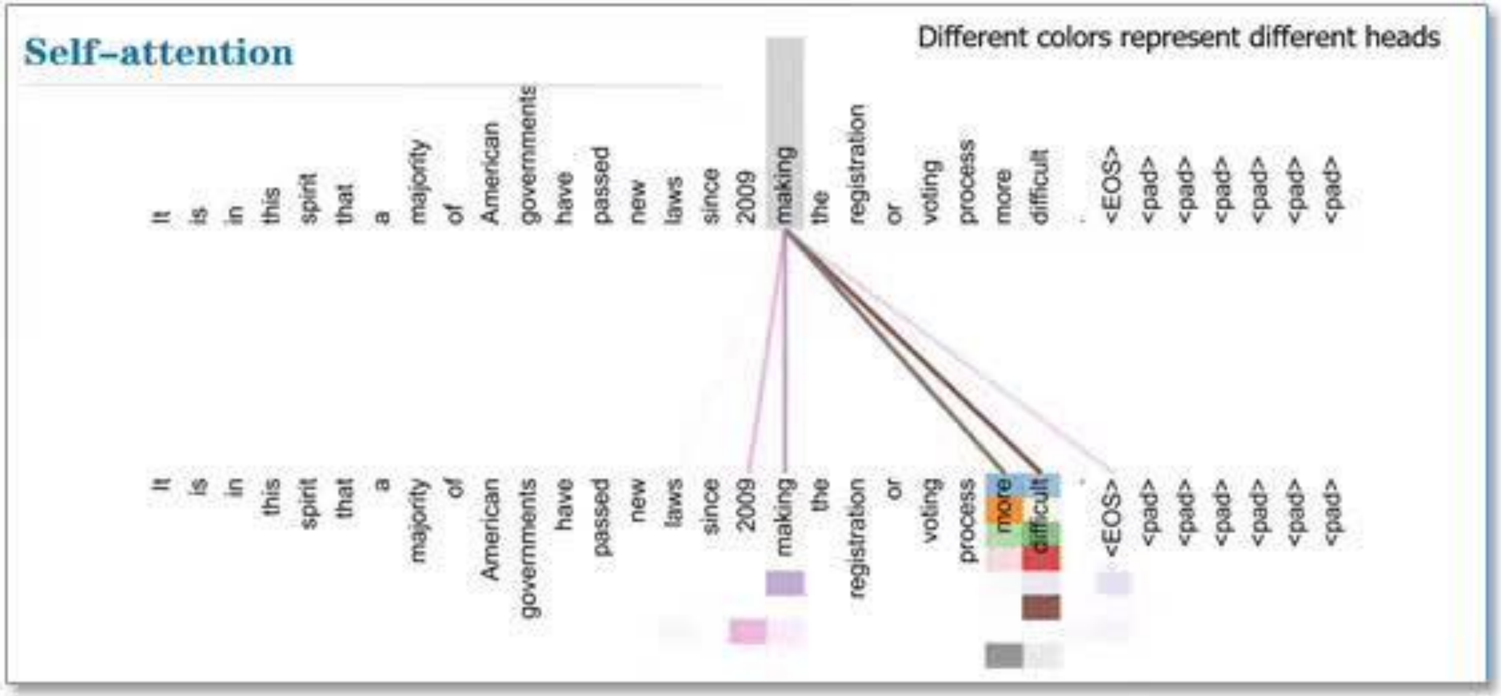
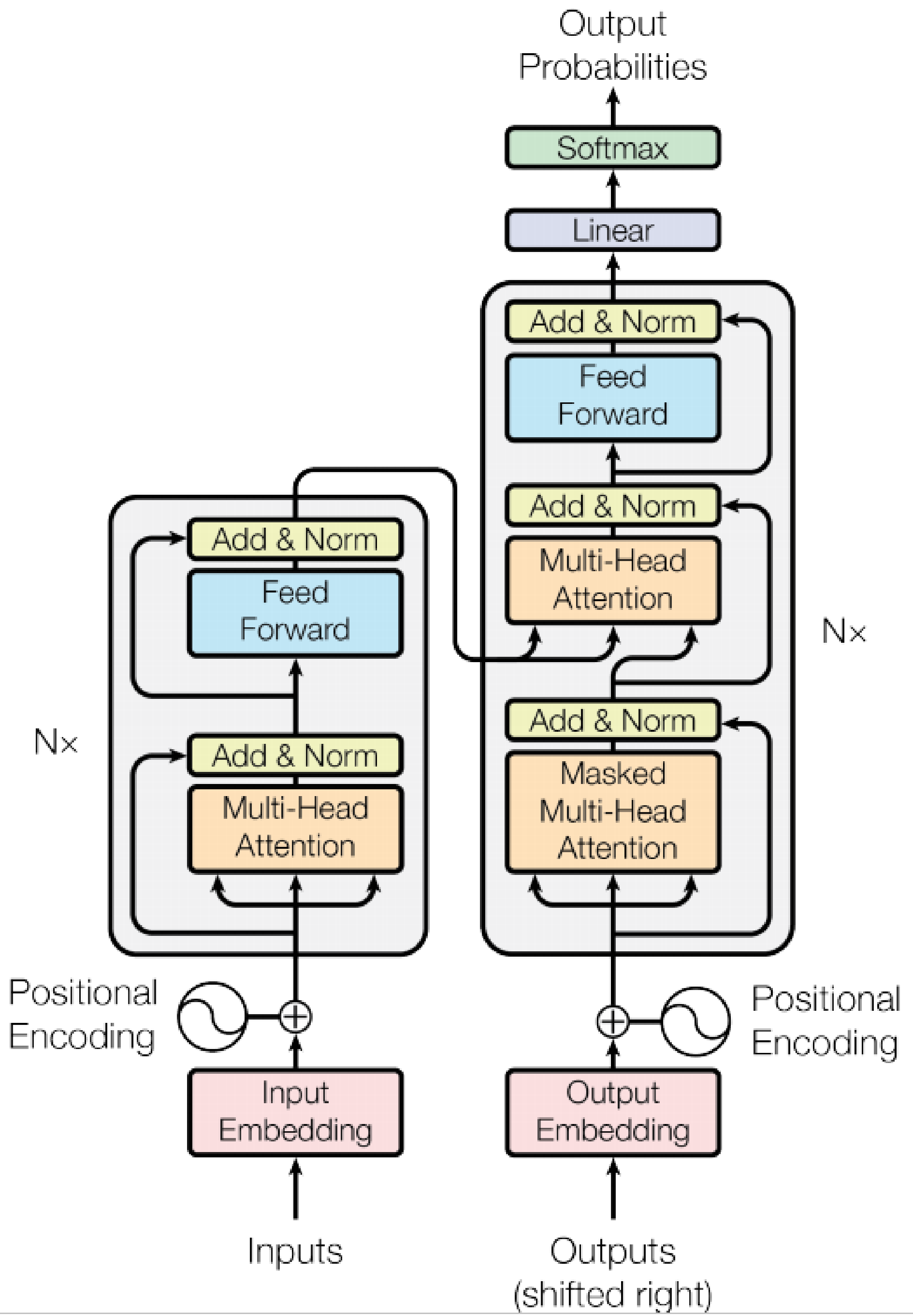


Fig. 6. The current word is in red and the size of the blue shade indicates the activation level. (Image source: Cheng et al., 2016)

Self-attention（自注意力机制）

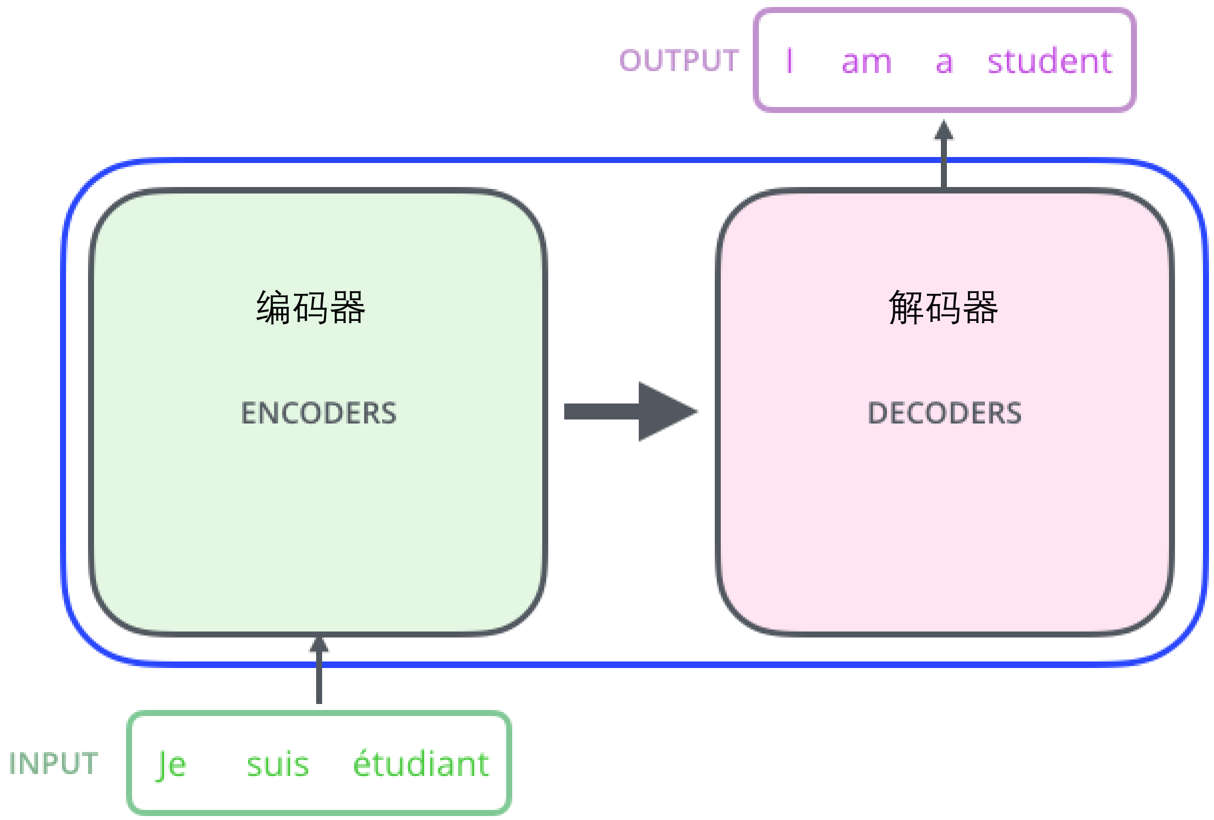


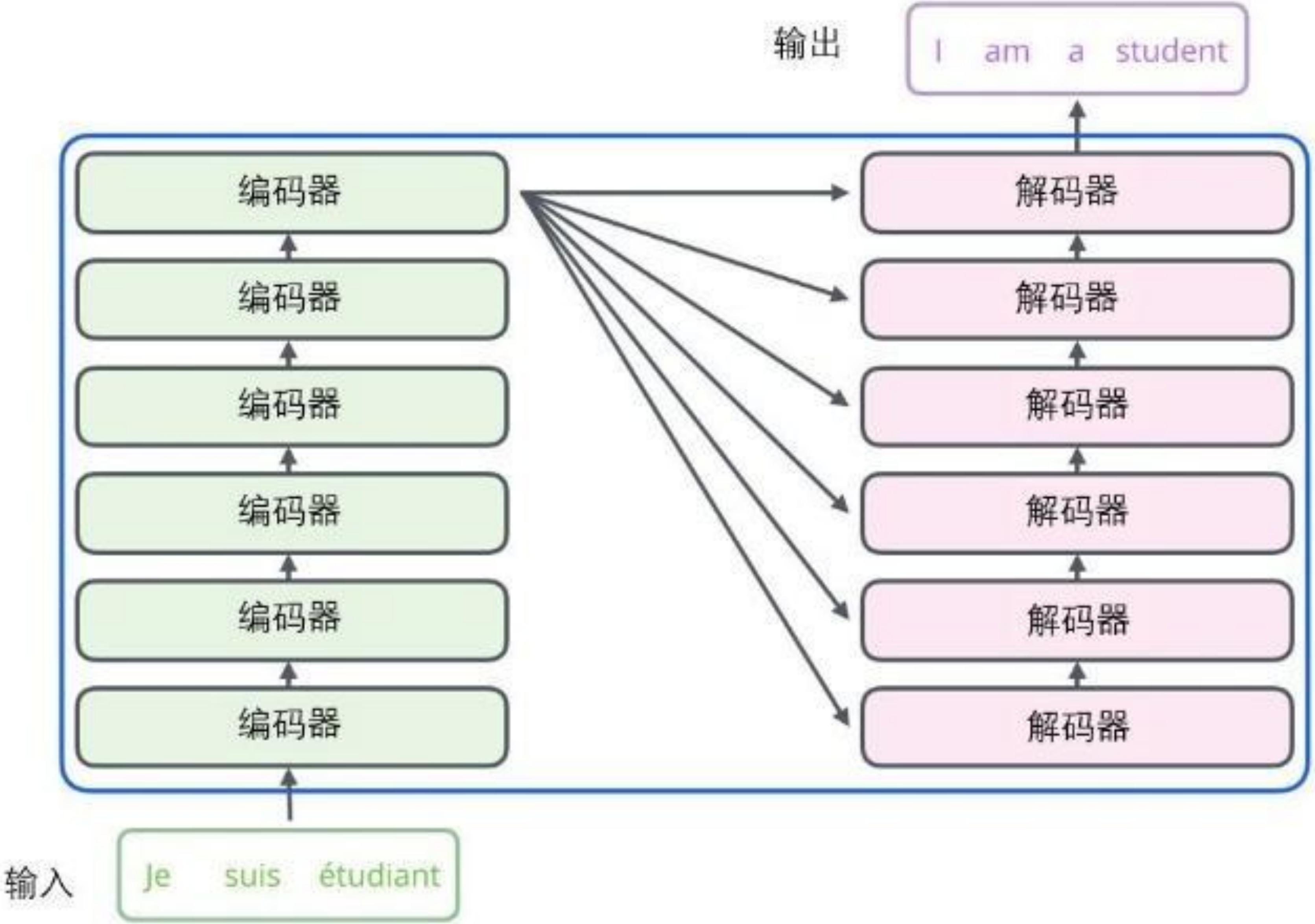
新的统治者!

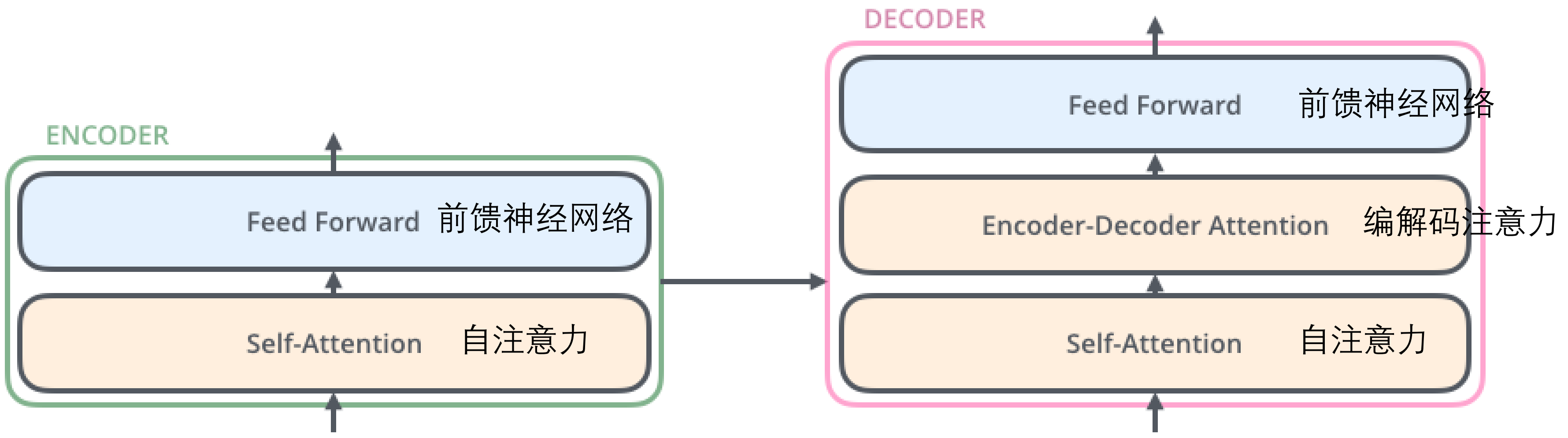


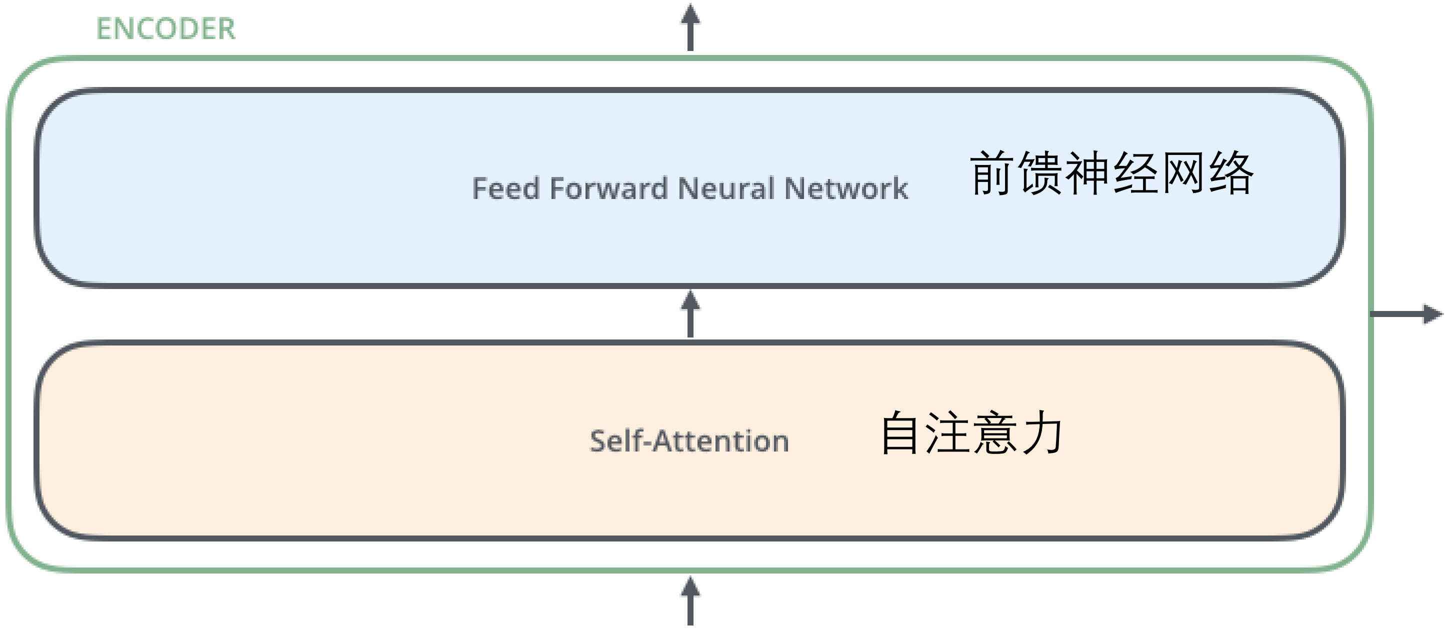
Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Add & Norm
Masked
Multi-Head
Attention
Nx
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Positional
Encoding
Positional
Encoding
Input
Embedding
Output
Embedding
Inputs
Outputs
(shifted right)

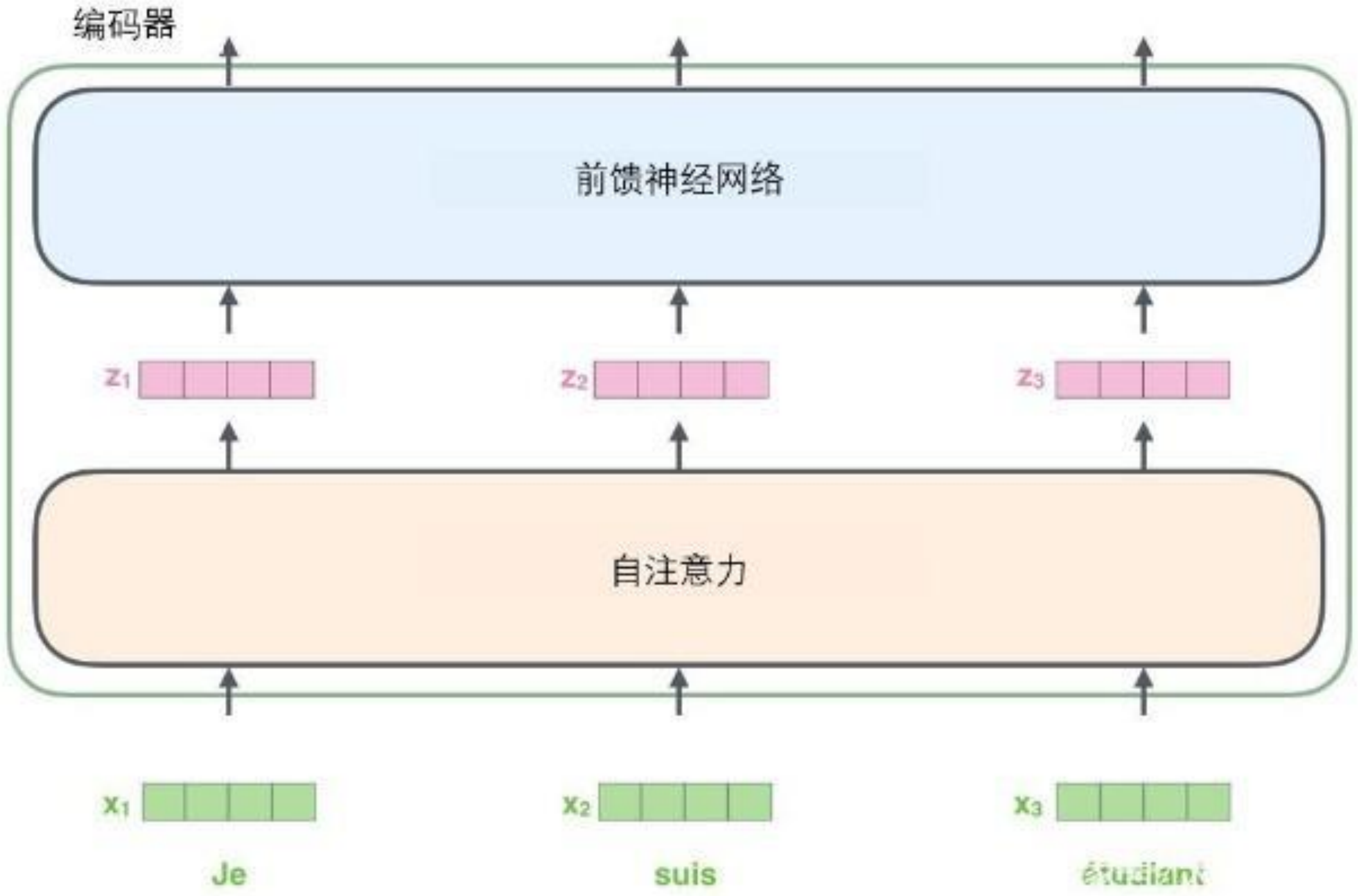

INPUT
Je suis étudiant
THE
TRANSFORMER
OUTPUT
I am a student

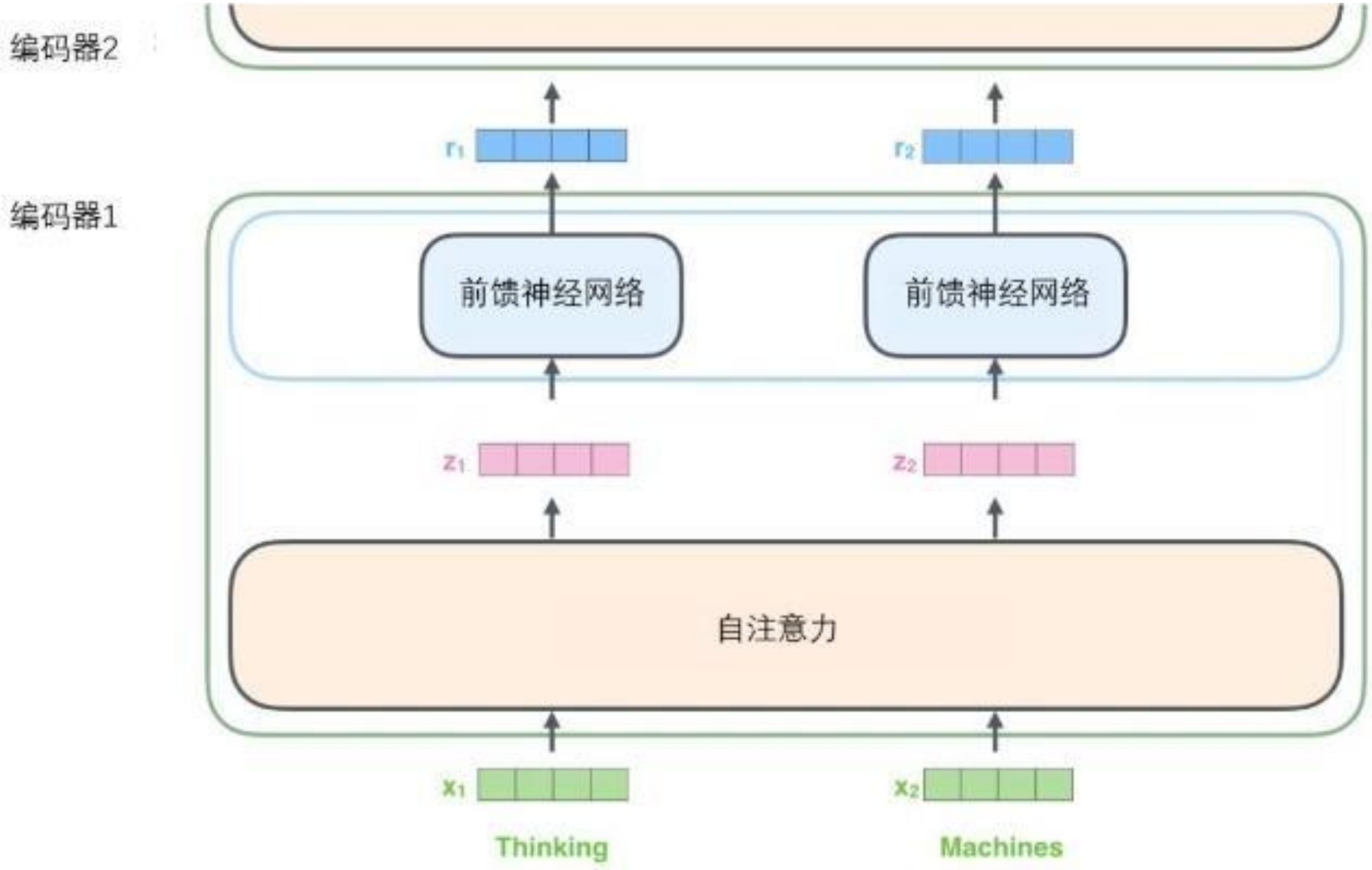

OUTPUT
I am a student
编码器
ENCODERS
解码器
DECODERS
INPUT
Je suis étudiant


输出
I am a student
编码器
编码器
编码器
编码器
编码器
编码器
解码器
解码器
解码器
解码器
解码器
解码器
输入
Je suis étudiant


ENCODER
Feed Forward 前馈神经网络
Self-Attention 自注意力
DECODER
Feed Forward 前馈神经网络
Encoder-Decoder Attention 编解码注意力
Self-Attention 自注意力

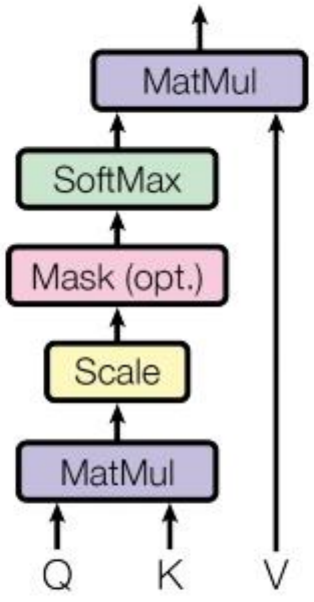

ENCODER
Feed Forward Neural Network 前馈神经网络
Self-Attention 自注意力


编码器
前馈神经网络
z1
z2
z3
自注意力
x1
x2
x3
Je
suis
étudiant

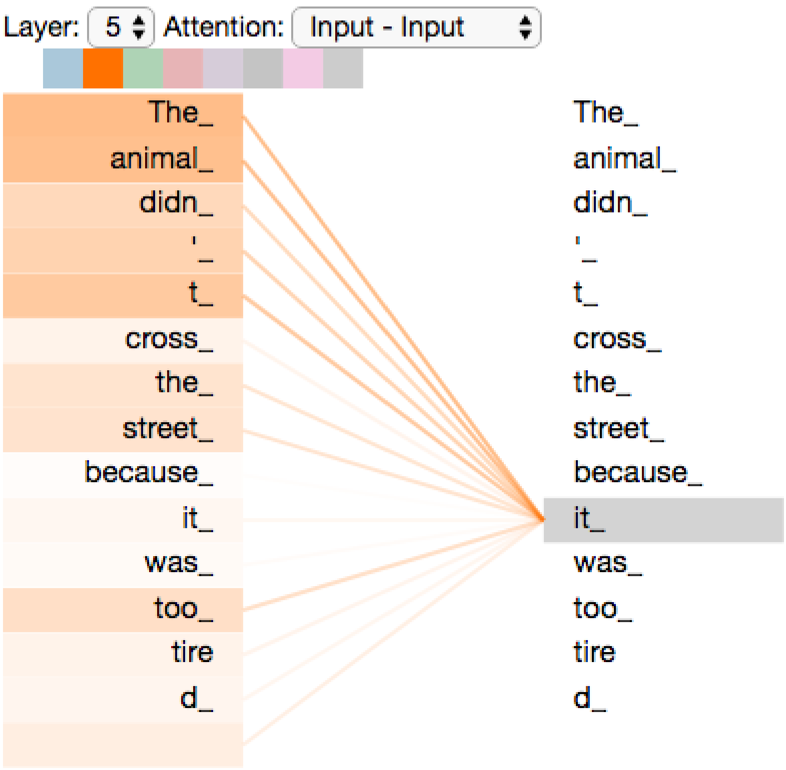

编码器2
r1
r2
编码器1
前馈神经网络
前馈神经网络
z1
z2
自注意力
x1
x2
Thinking
Machines

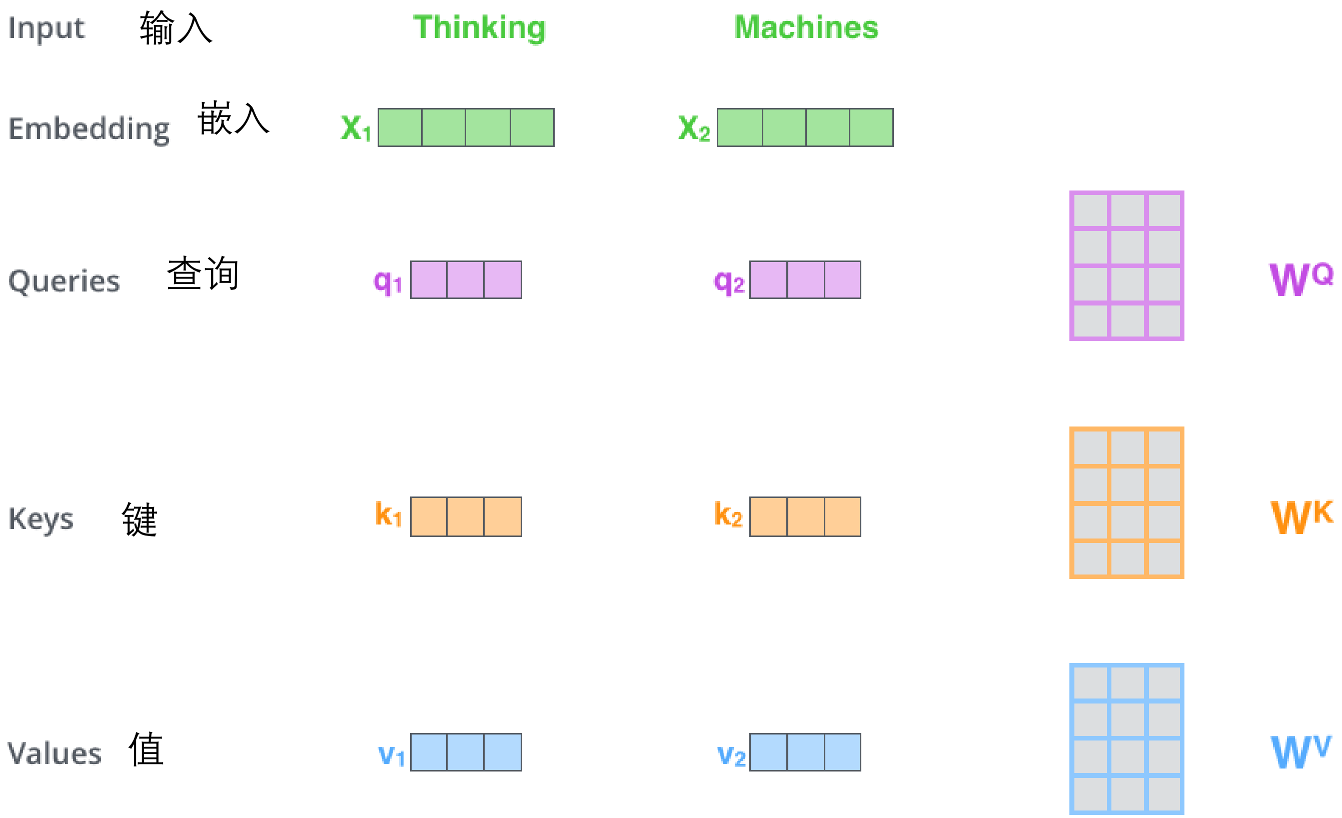
降尺度的点乘注意力机制

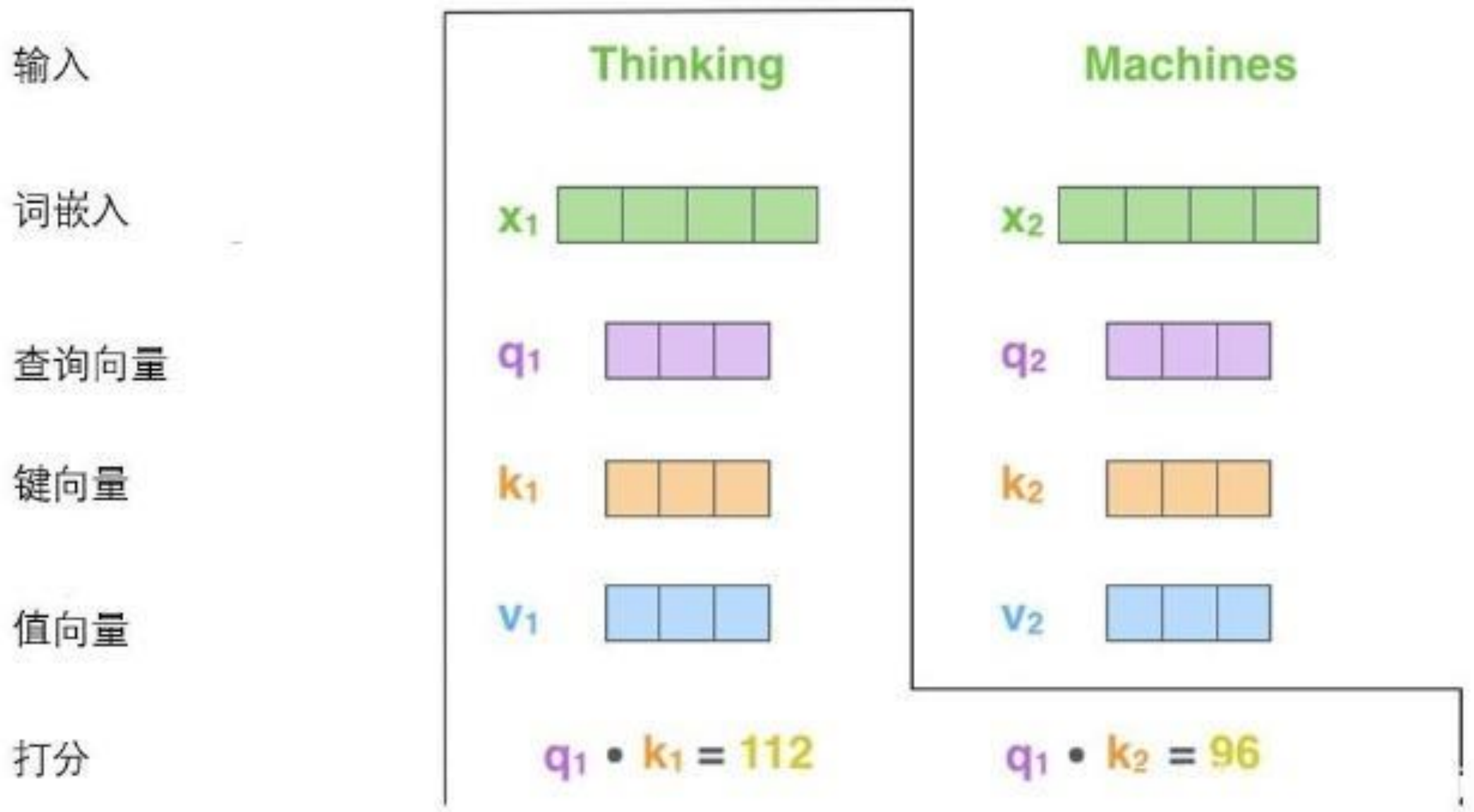
Scaled Dot-Product Attention

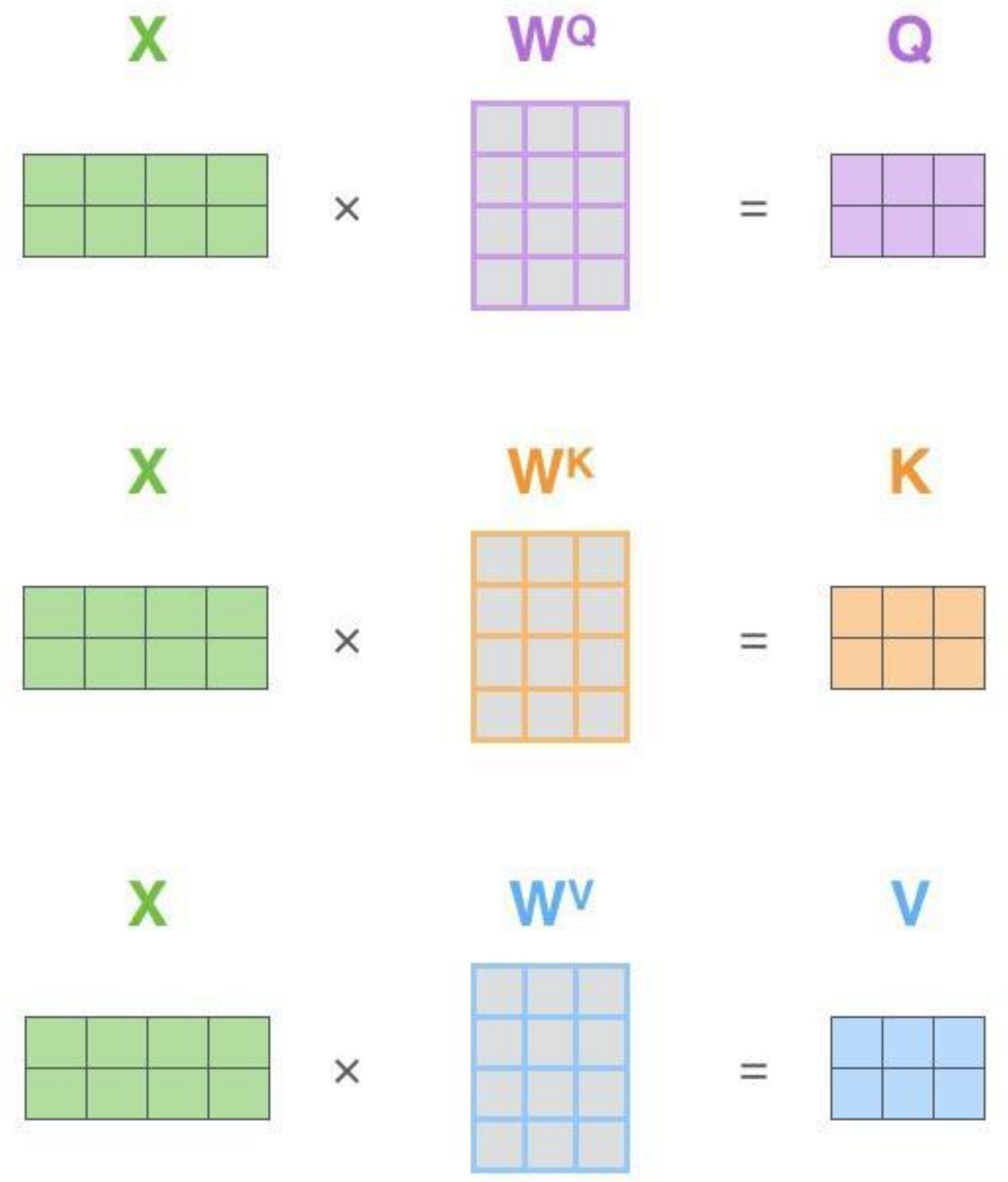


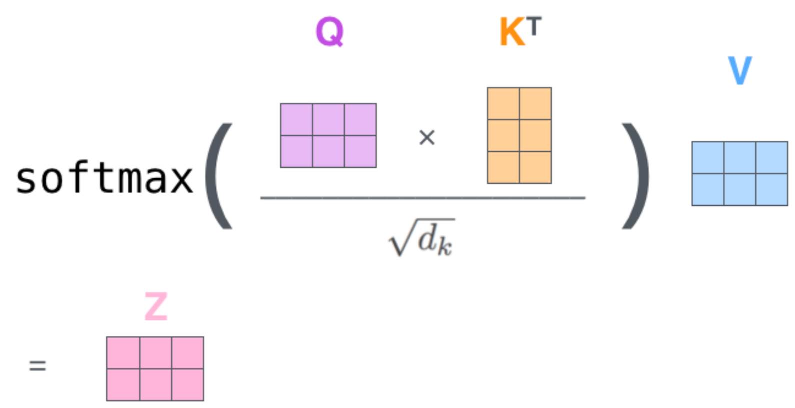
$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

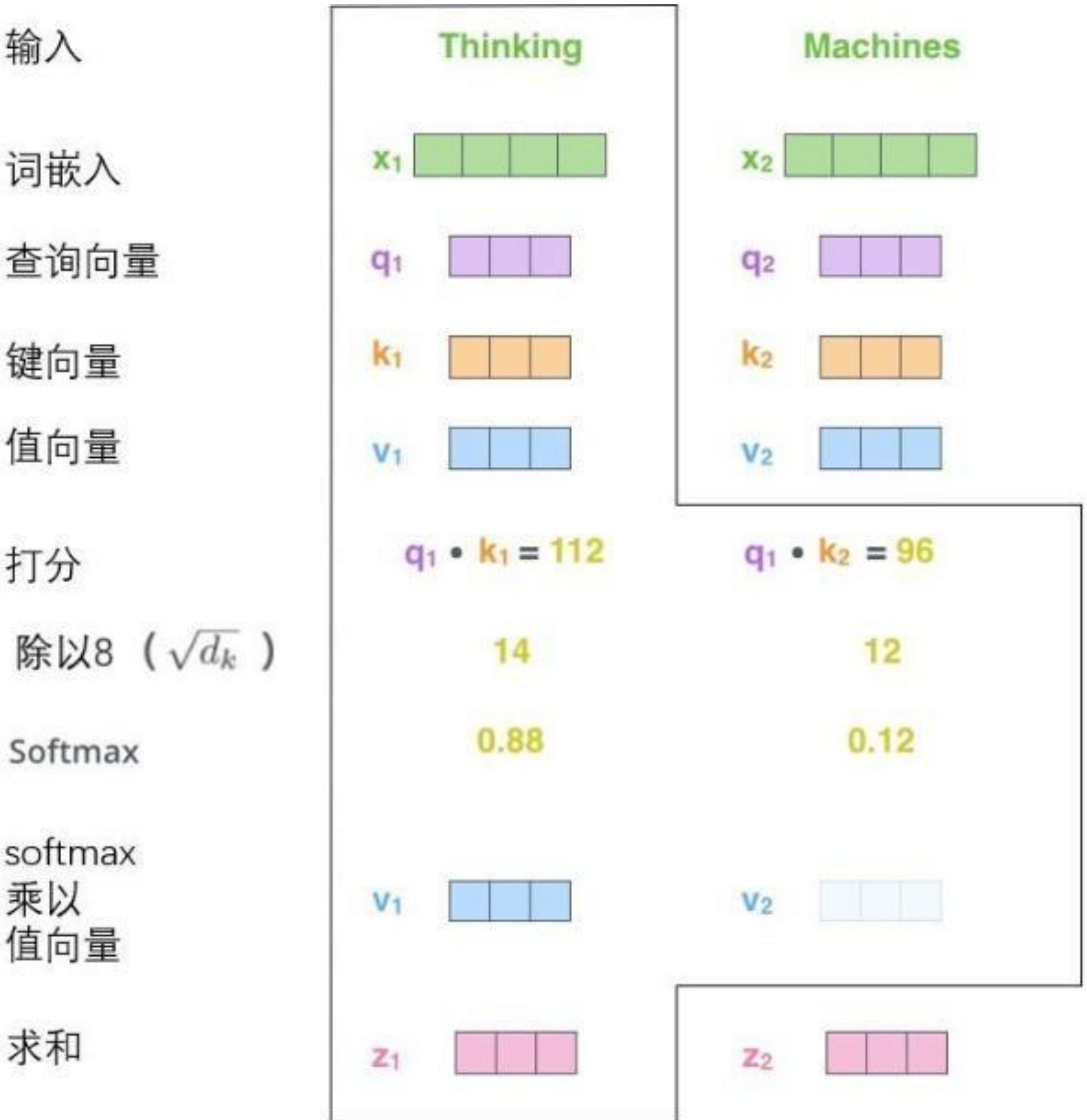

Layer: 5 Attention: Input - Input
The_
animal_
didn_
'_
t_
cross_
the_
street_
because_
it_
was_
too_
tire
d_


Input 输入
Thinking
Machines
Embedding 嵌入
x_1
x_2
Queries 查询
q_1
q_2
W^Q
Keys 键
k_1
k_2
W^K
Values 值
v_1
v_2
W^V


输入
词嵌入
查询向量
键向量
值向量
打分
Thinking
Machines
x_1
x_2
q_1
q_2
k_1
k_2
v_1
v_2
$q_1 \bullet k_1 = 112$
$q_1 \bullet k_2 = 96$

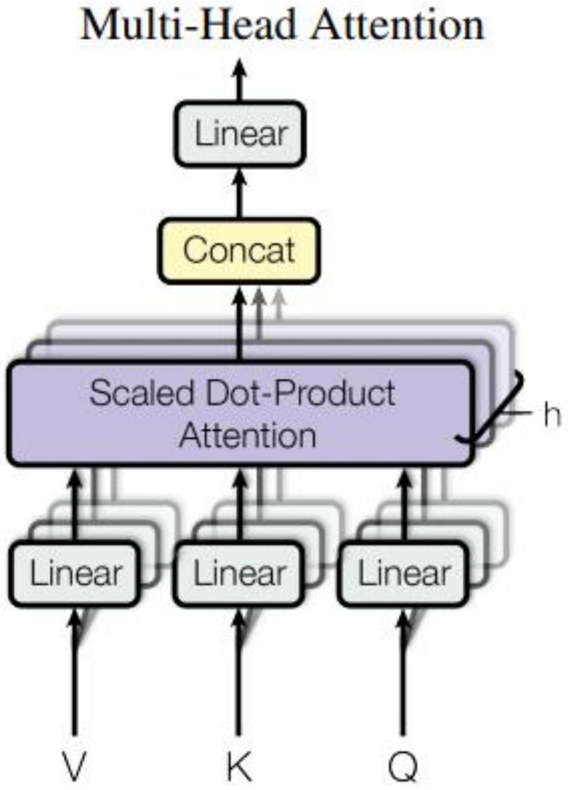

X
W^Q
Q
×
=
X
W^K
K
×
=
X
W^V
V
×
=


Q
K^T
V
softmax
×
√d_k
=
Z

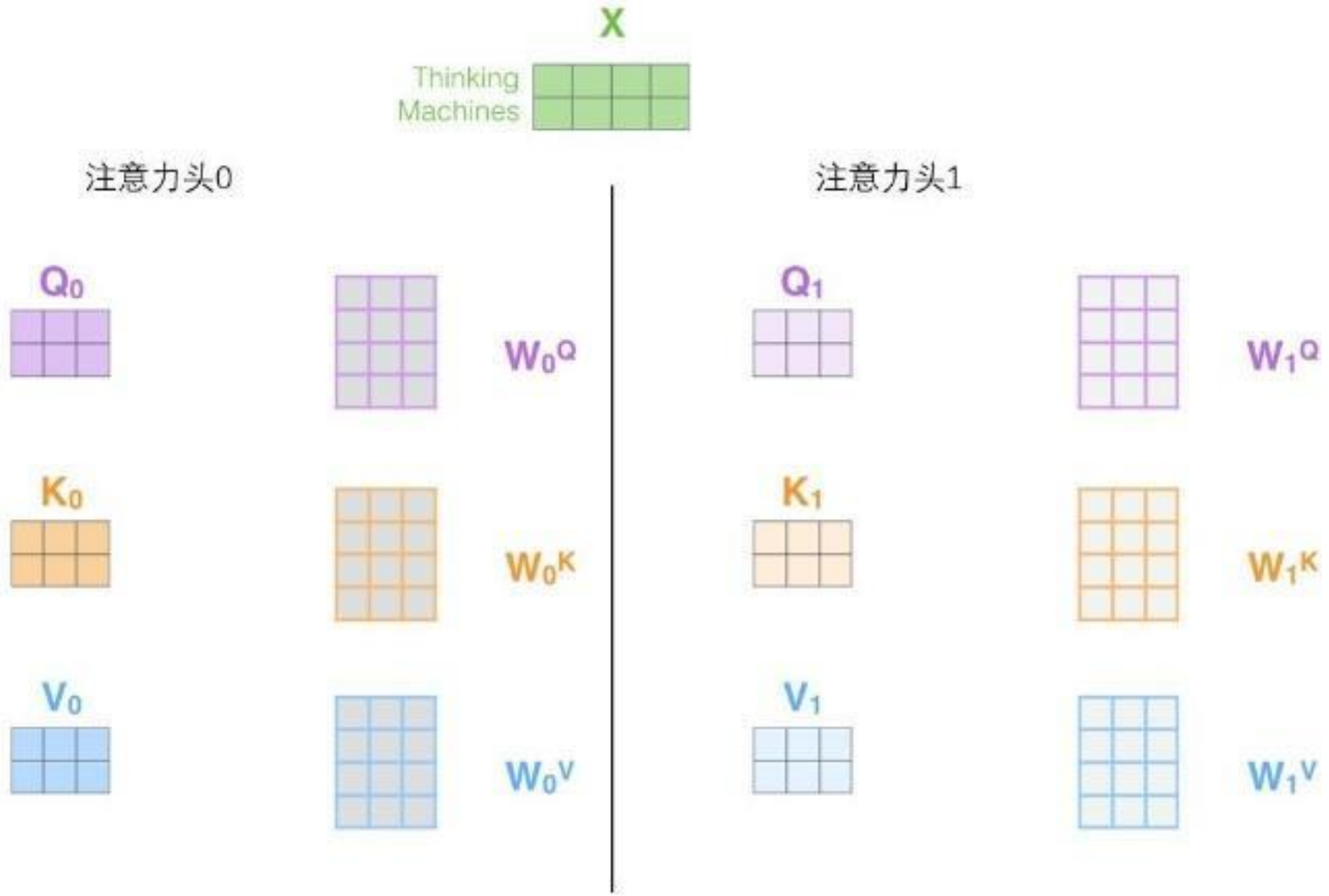

输入
Thinking
Machines
词嵌入
x_1
x_2
查询向量
q_1
q_2
键向量
k_1
k_2
值向量
v_1
v_2
打分
$q_1 \cdot k_1 = 112$
$q_1 \cdot k_2 = 96$
除以8（$\sqrt{d_k}$）
14
12
Softmax
0.88
0.12
softmax
乘以
值向量
v_1
v_2
求和
z_1
z_2

多头注意力机制


CHO'GALL
SPOTLIGHT

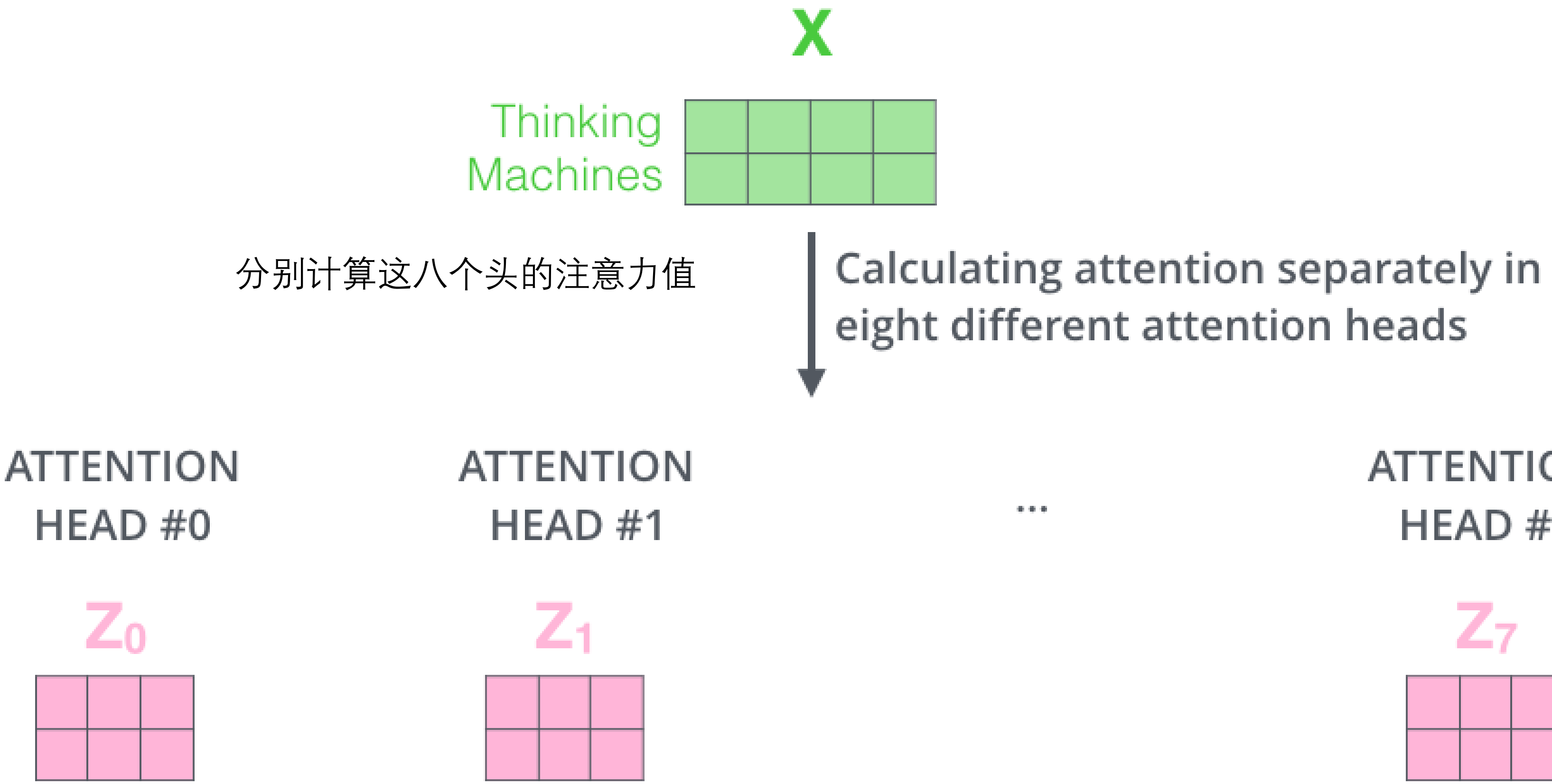

Multi-Head Attention
Linear
Concat
Scaled Dot-Product
Attention
h
Linear
Linear
Linear
V
K
Q

多头注意力机制



X
Thinking
Machines
注意力头0
注意力头1
Q_0
W_0^Q
K_0
W_0^K
V_0
W_0^V
Q_1
W_1^Q
K_1
W_1^K
V_1
W_1^V

多头注意力机制

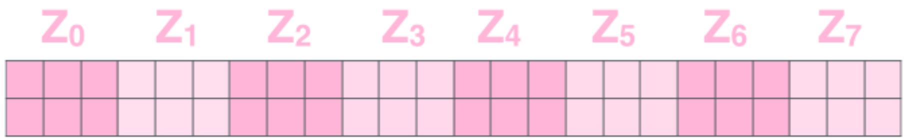

X
Thinking
Machines
分别计算这八个头的注意力值
Calculating attention separately in
eight different attention heads
ATTENTION
HEAD #0
Z_0
ATTENTION
HEAD #1
Z_1
...
ATTENTION
HEAD #7
Z_7

多头注意力机制

$$\mathrm{MultiHead}(Q,K,V) = \mathrm{Concat}(\mathrm{head_1},...,\mathrm{head_h})W^O$$
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

将所有的注意力头拼接

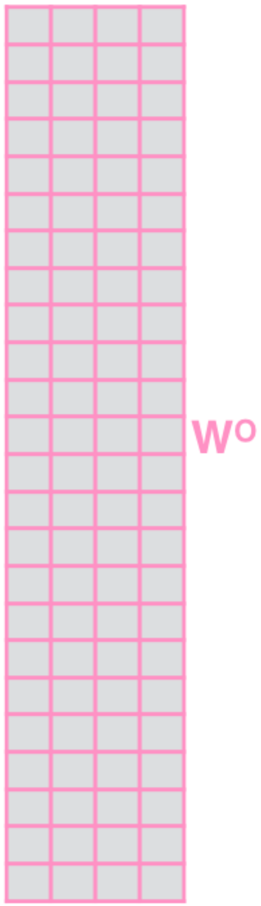
1) Concatenate all the attention heads



与W0相乘，这个矩阵也会和模型一起训练

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

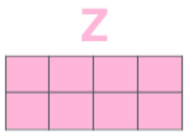
X

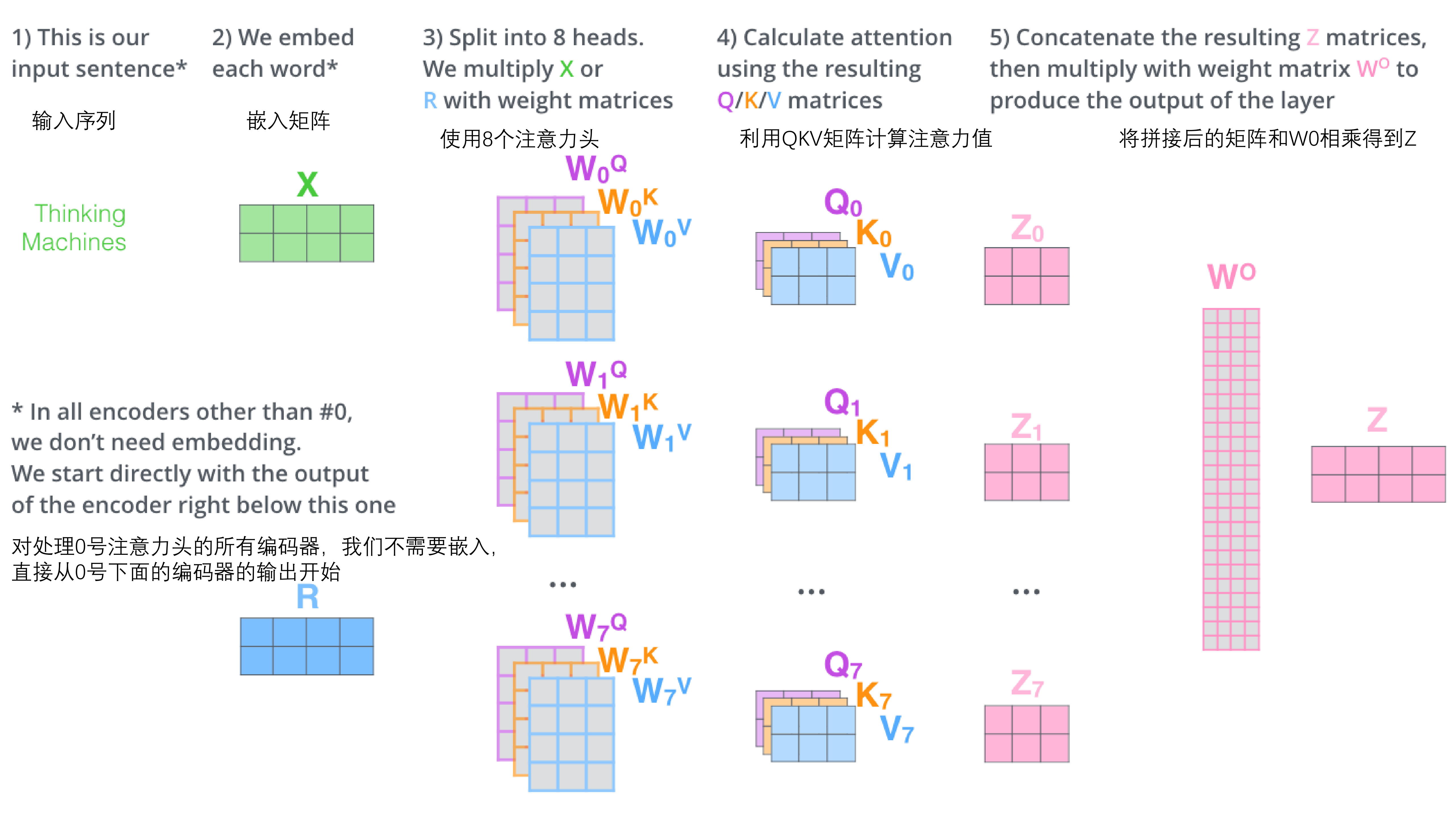


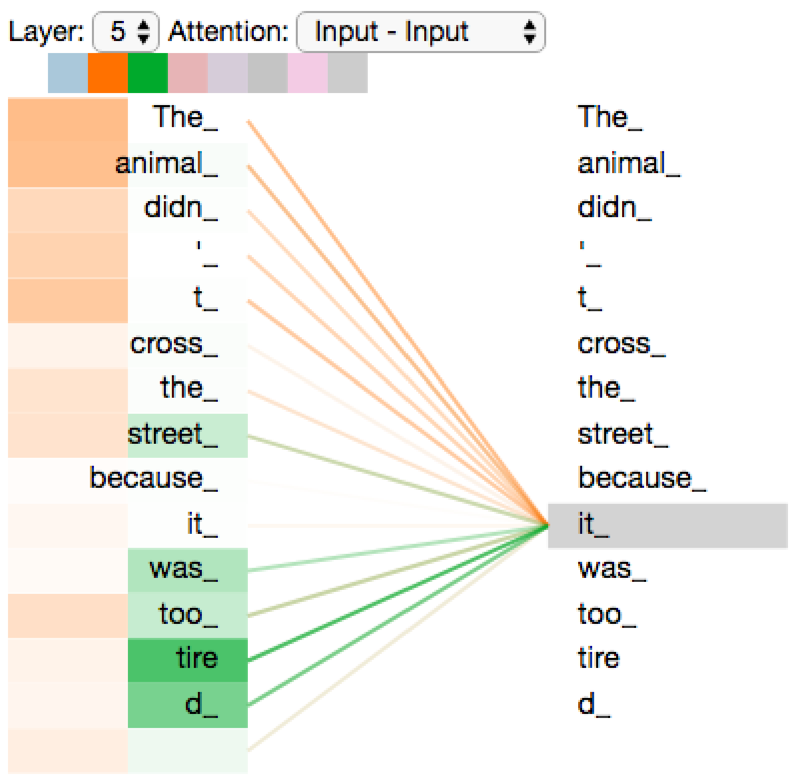
最后的Z矩阵会捕捉所有注意力头的特征，然后讲这个Z送往前馈网络。

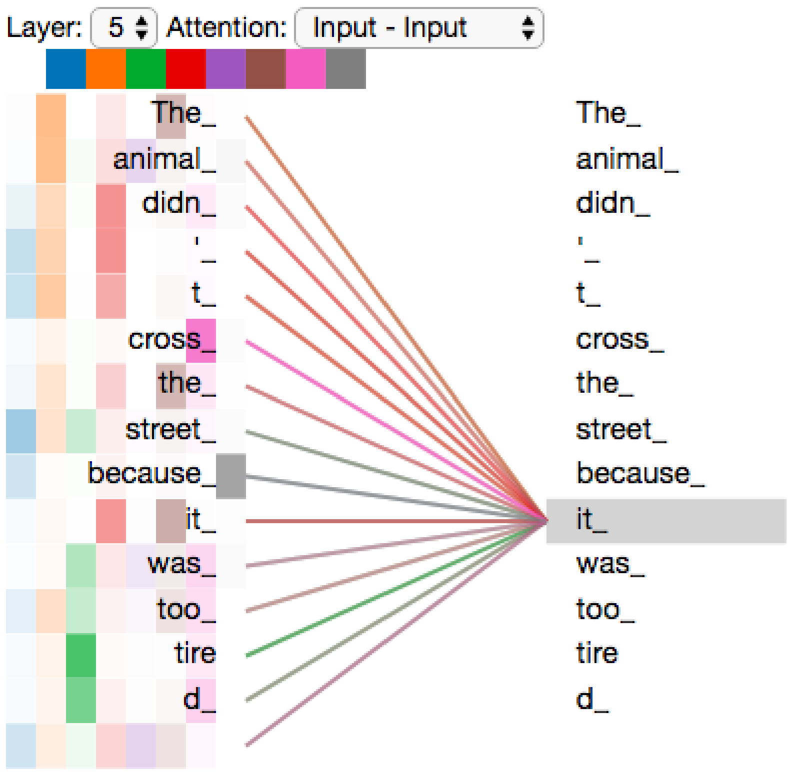
3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN

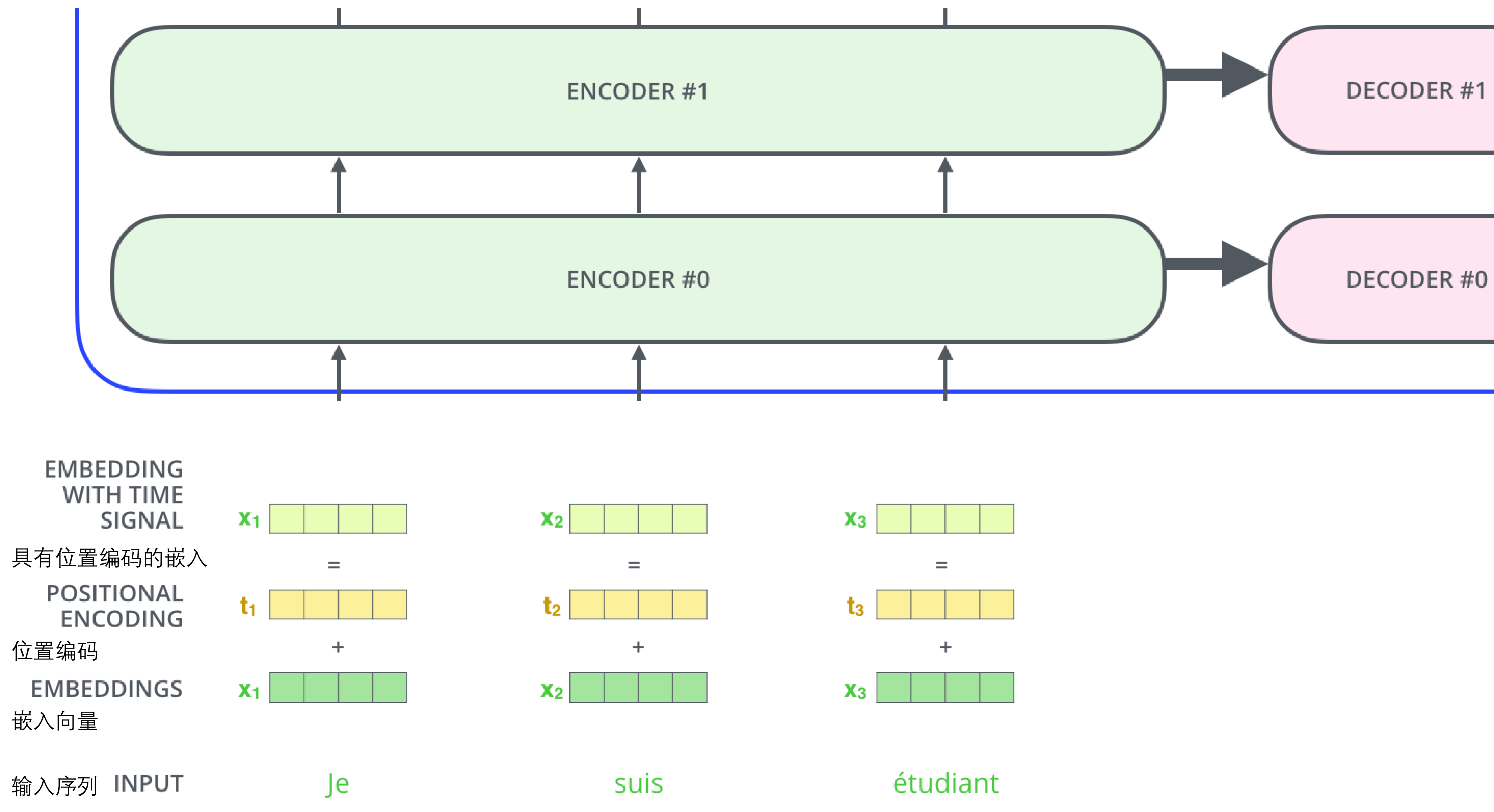
=



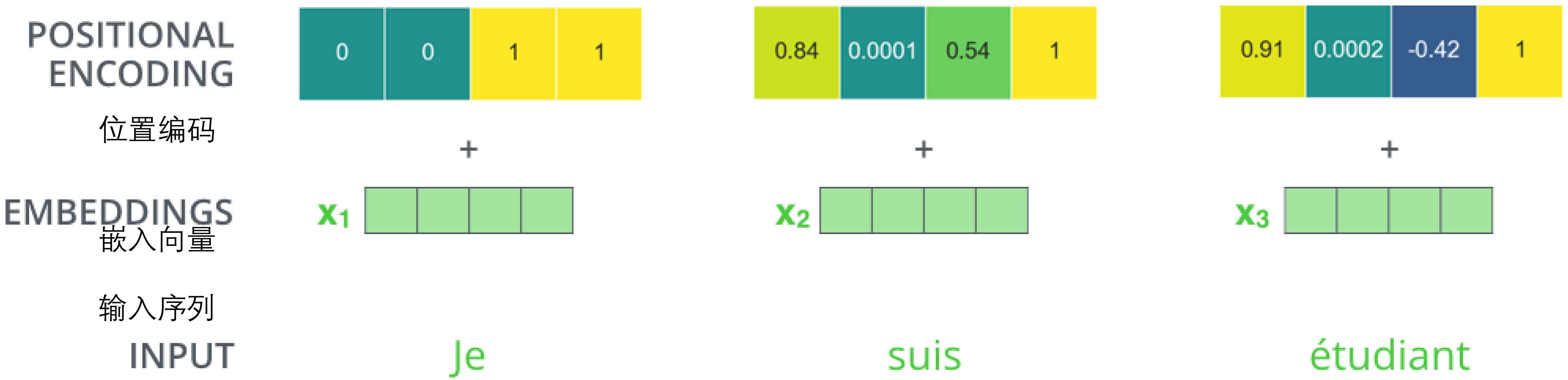

1) This is our input sentence*
输入序列
2) We embed each word*
嵌入矩阵
3) Split into 8 heads. We multiply X or R with weight matrices
使用8个注意力头
4) Calculate attention using the resulting Q/K/V matrices
利用QKV矩阵计算注意力值
5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer
将拼接后的矩阵和W0相乘得到Z
Thinking
Machines
X
W0Q
W0K
W0V
Q0
K0
V0
Z0
WO
W1Q
W1K
W1V
Q1
K1
V1
Z1
Z
* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one
对处理0号注意力头的所有编码器，我们不需要嵌入，
直接从0号下面的编码器的输出开始
R
W7Q
W7K
W7V
Q7
K7
V7
Z7

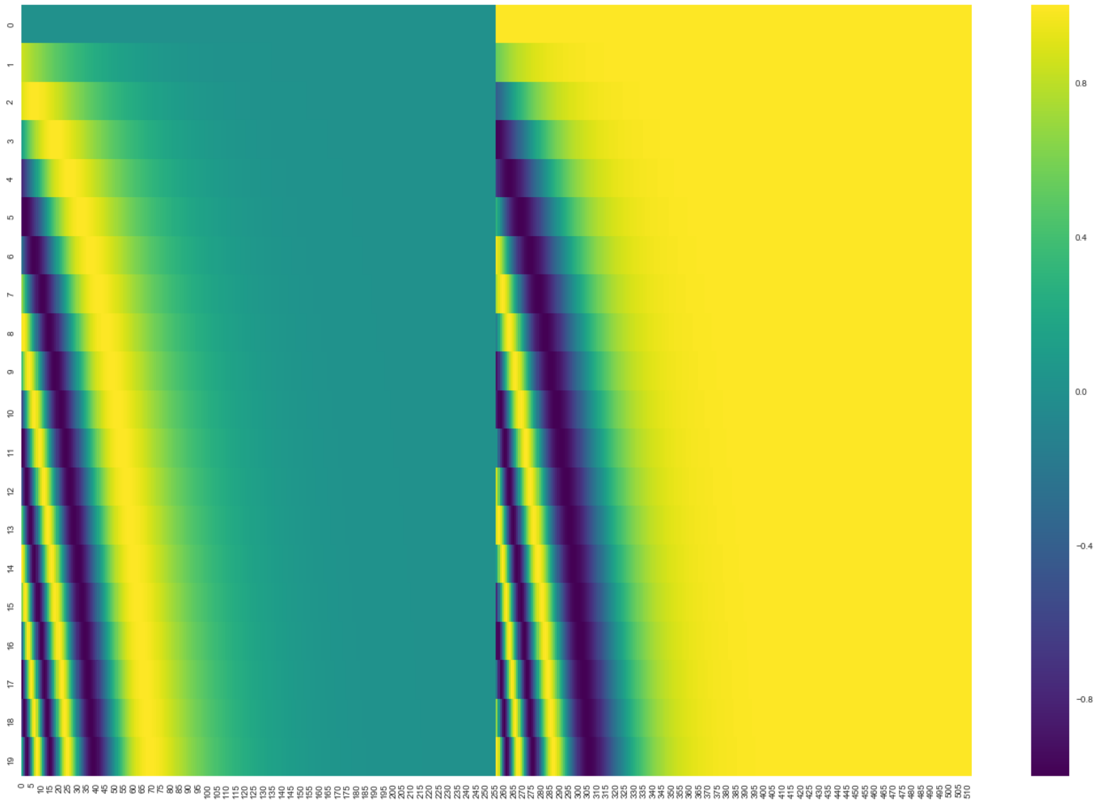

Layer: 5
Attention: Input - Input
The_
animal_
didn_
'_
t_
cross_
the_
street_
because_
it_
was_
too_
tire
d_
The_
animal_
didn_
'_
t_
cross_
the_
street_
because_
it_
was_
too_
tire
d_


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

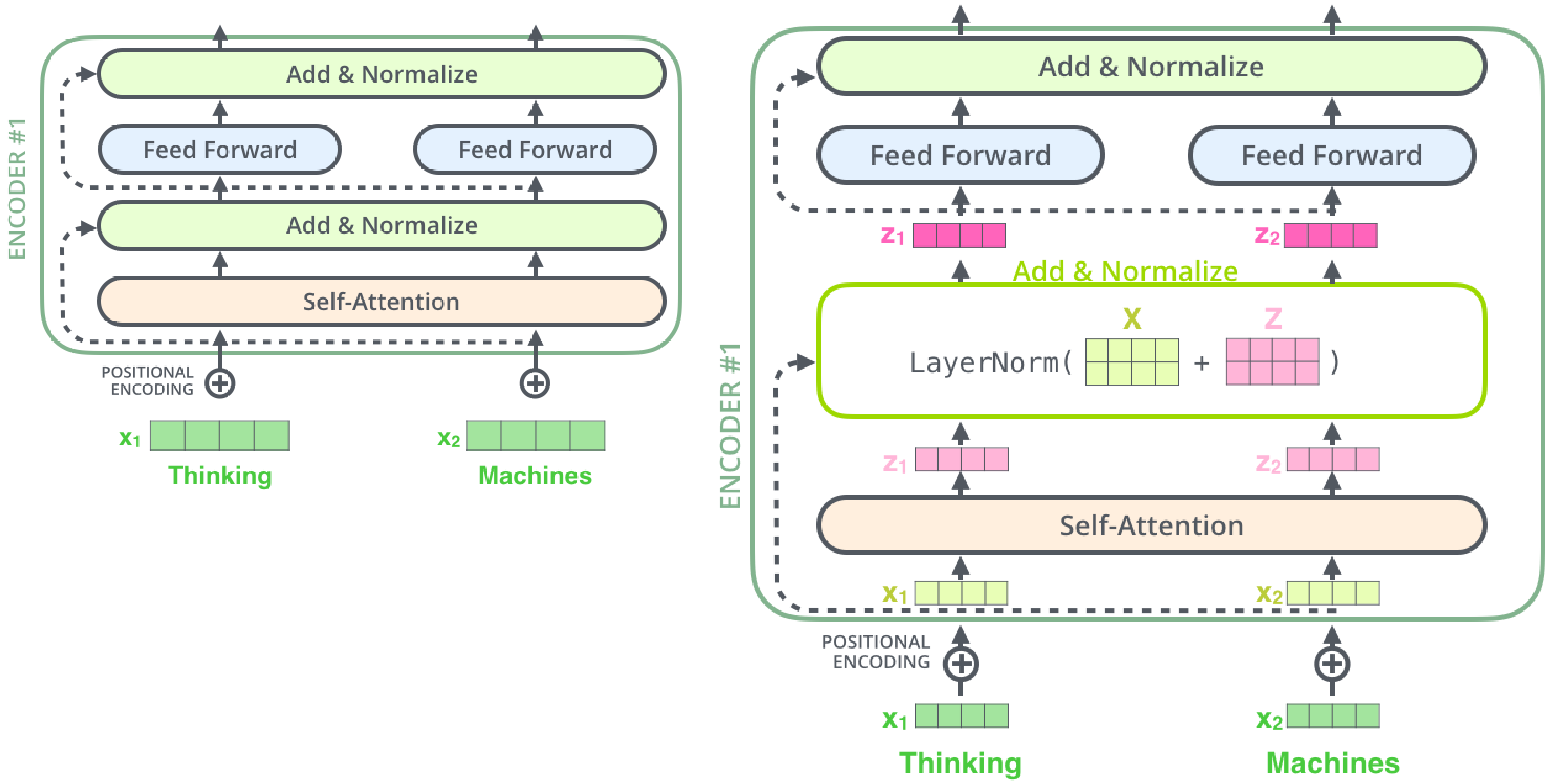


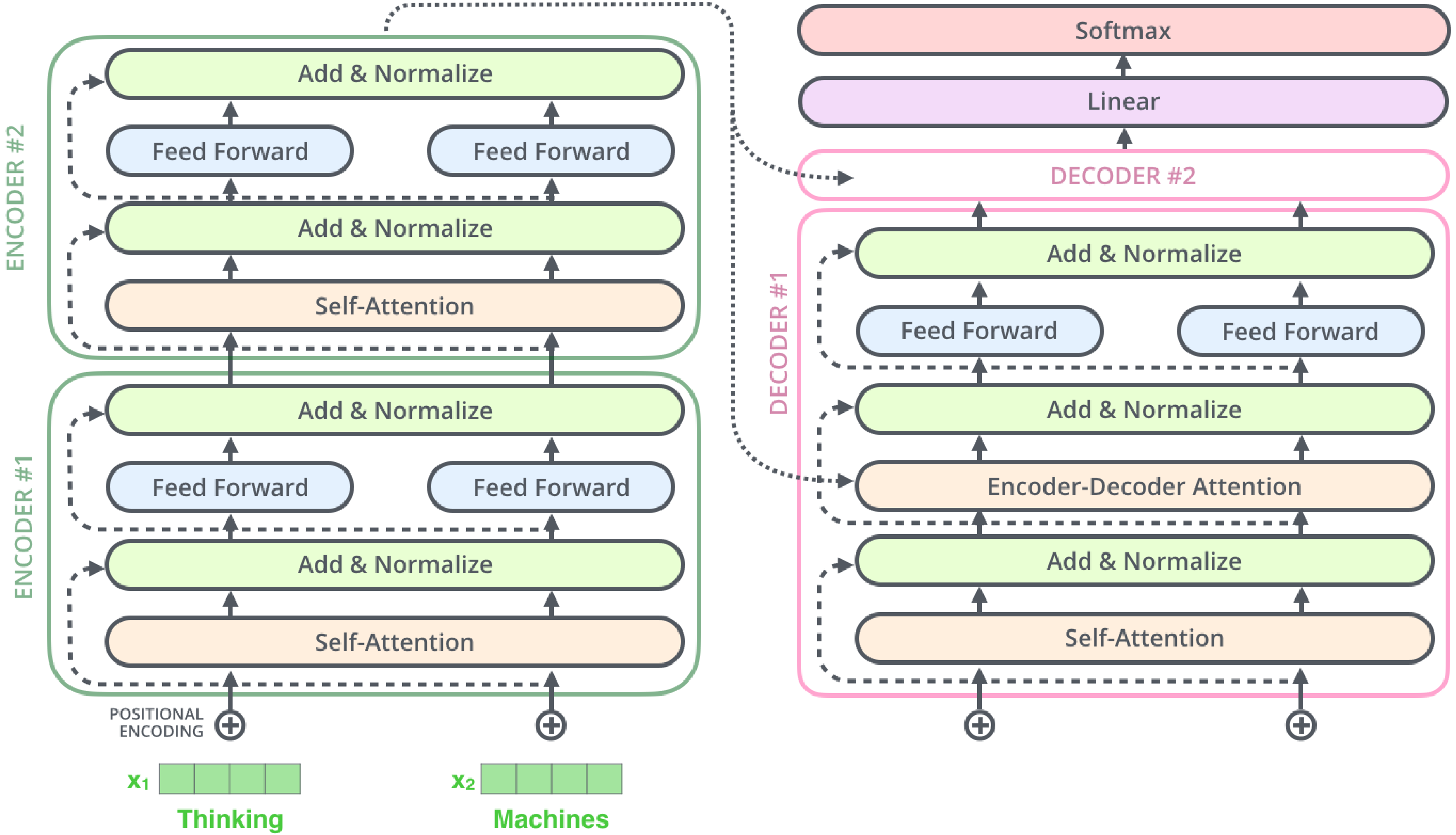
ENCODER #1
DECODER #1
ENCODER #0
DECODER #0
EMBEDDING WITH TIME SIGNAL
具有位置编码的嵌入
POSITIONAL ENCODING
位置编码
EMBEDDINGS
嵌入向量
输入序列 INPUT
x1
x2
x3
=
t1
t2
t3
+
Je
suis
étudiant

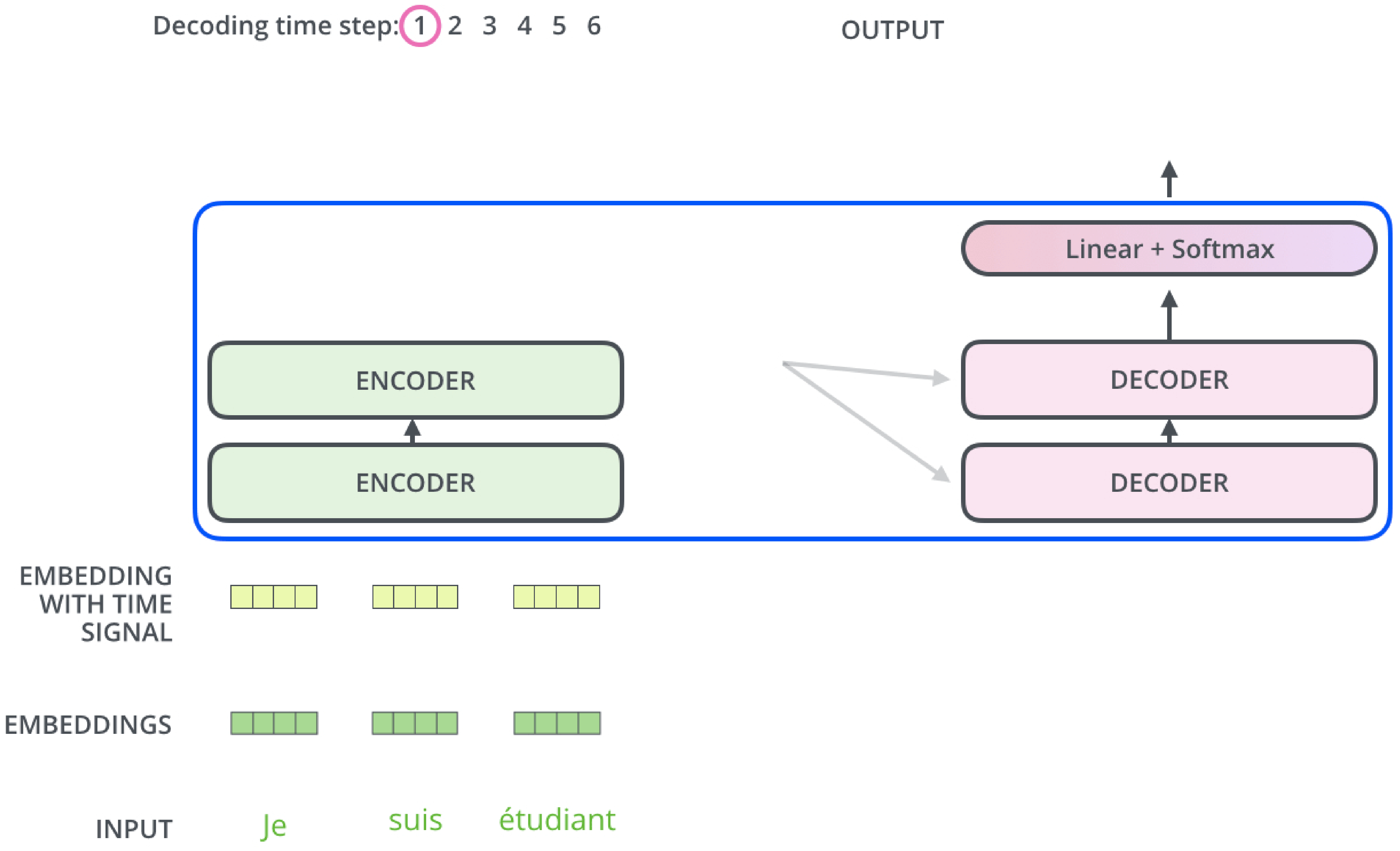


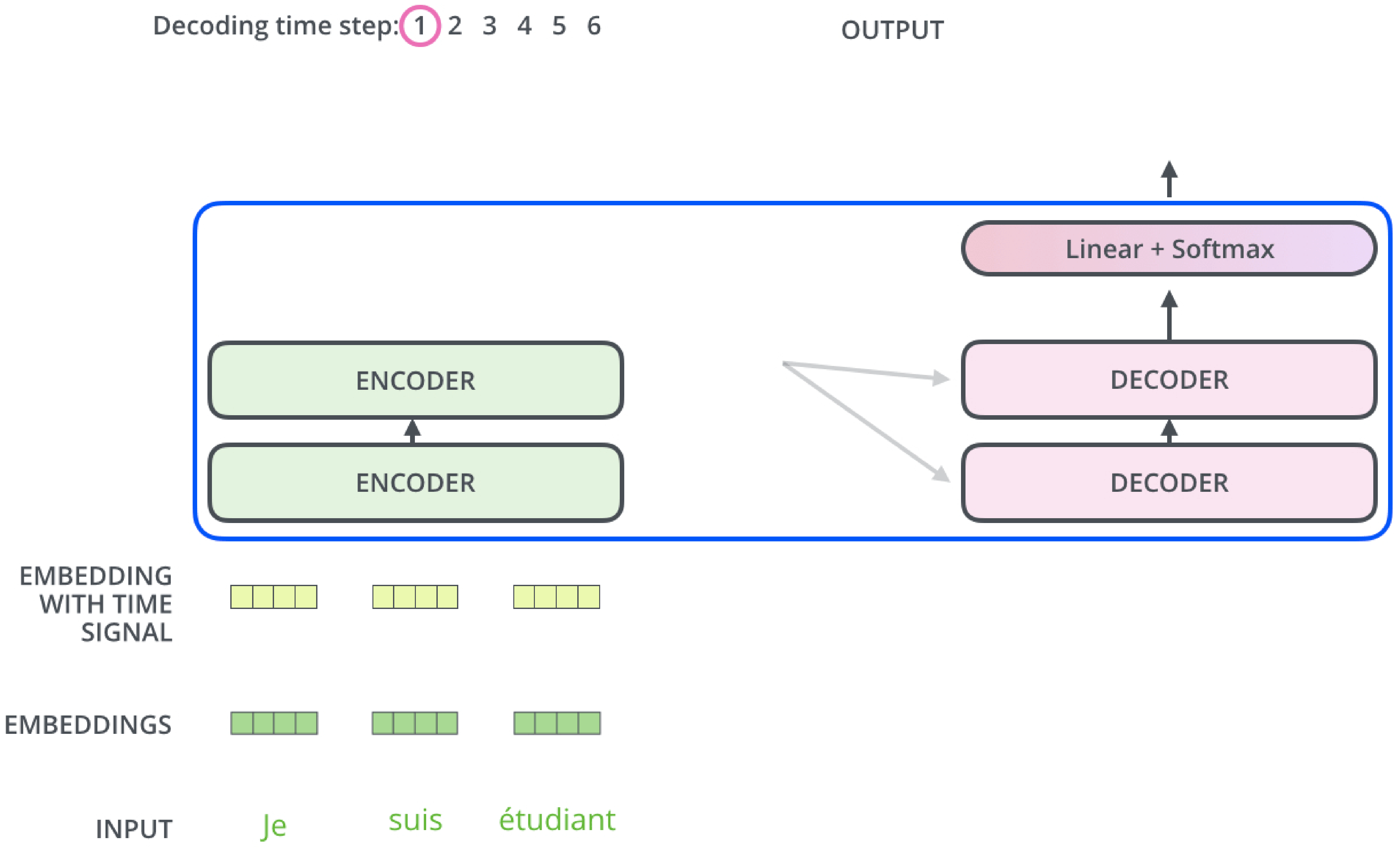


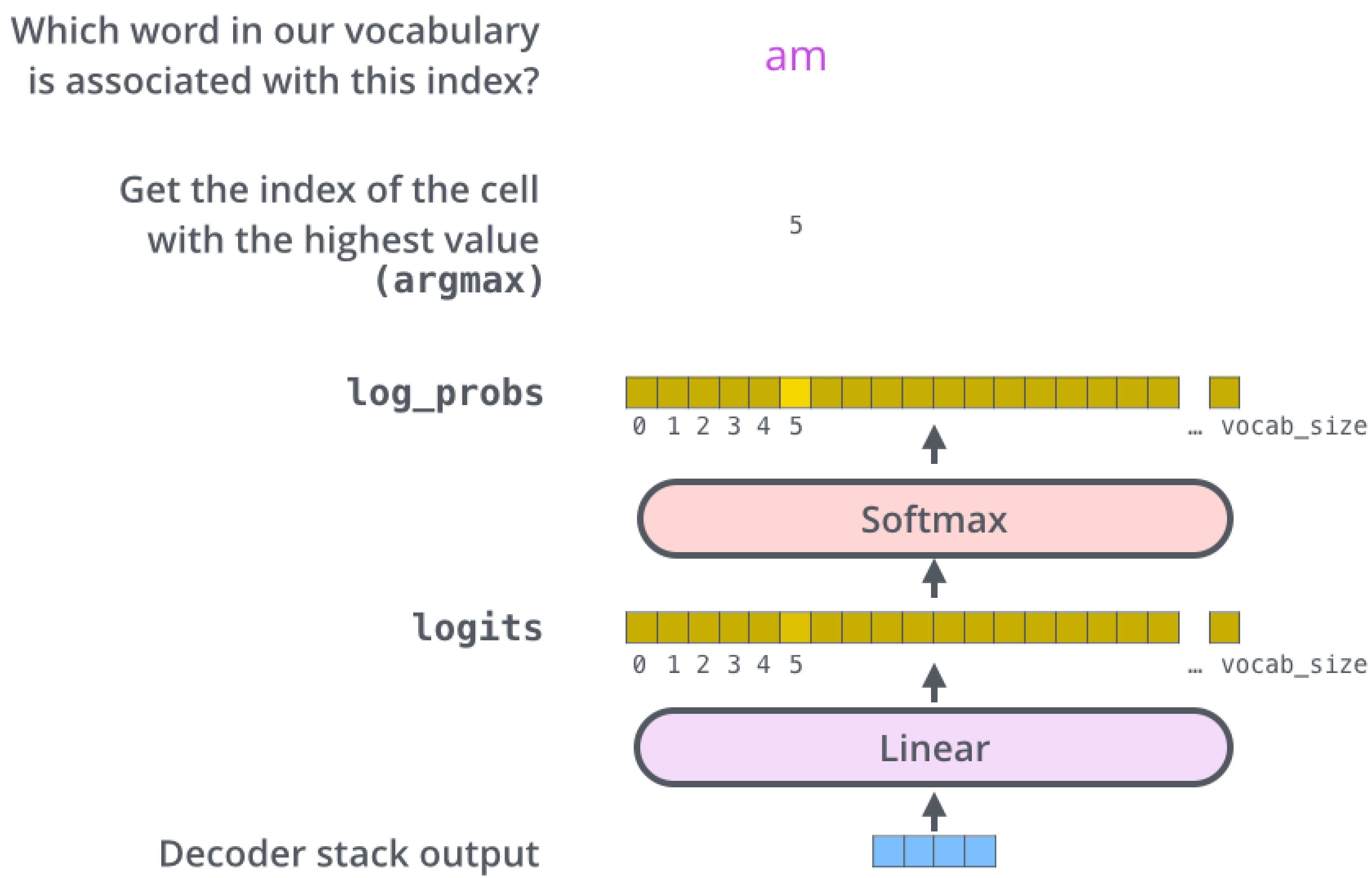
$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$


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


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
x1
x2
Thinking
Machines
Softmax
Linear
DECODER #2
DECODER #1
Add & Normalize
Feed Forward
Feed Forward
Add & Normalize
Encoder-Decoder Attention
Add & Normalize
Self-Attention


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
Linear + Softmax
ENCODER
ENCODER
DECODER
DECODER
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant


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