NLP项目

Transformer

课程内容

- Seq2 Seq结构和Attention结构回顾
- Transformer结构讲解

Seq2Seq结构回顾

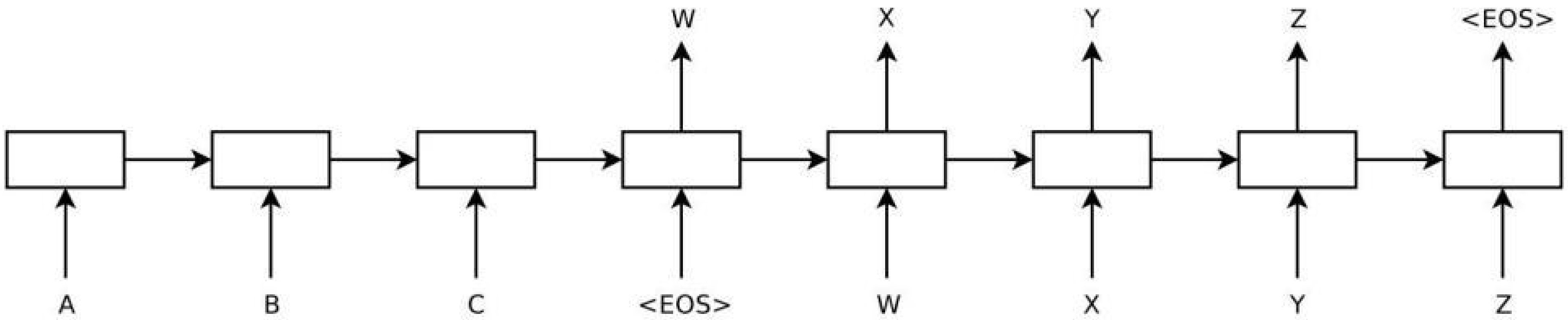
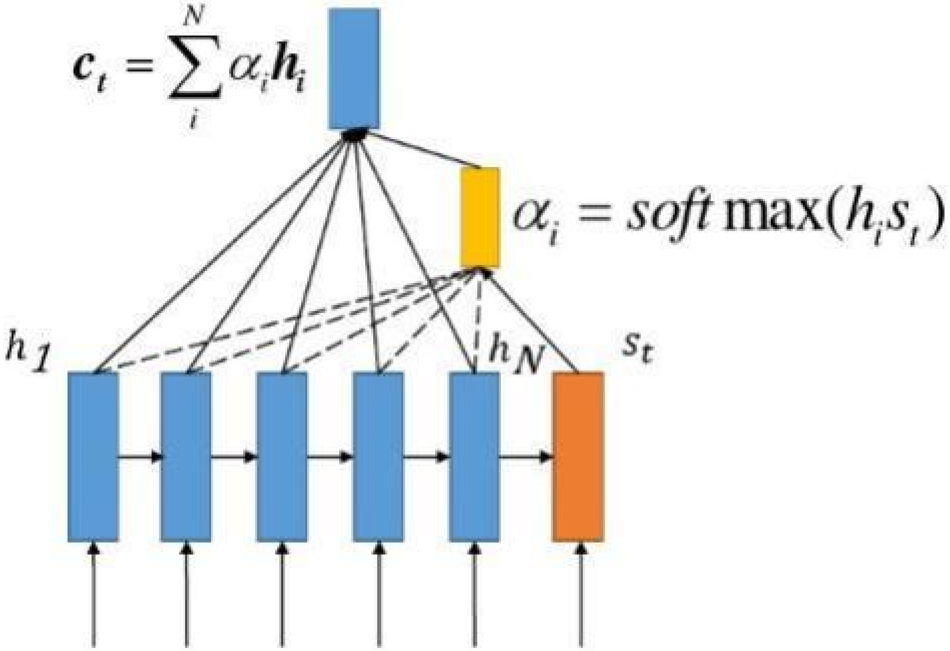
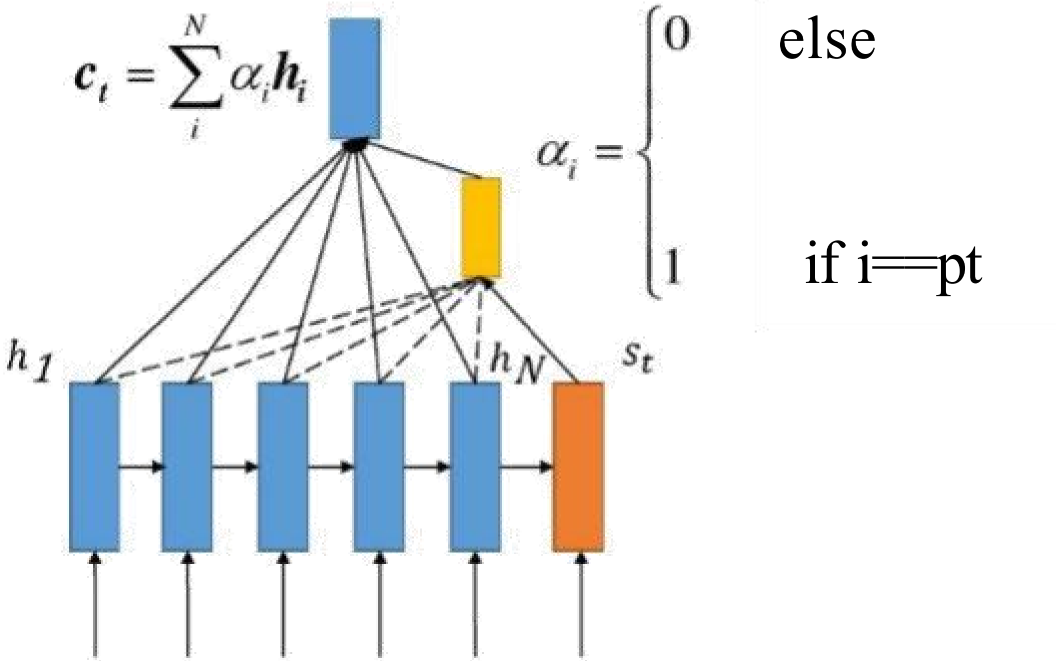


Figure 1: Our model reads an input sentence “ABC” and produces “WXYZ” as the output sentence. The model stops making predictions after outputting the end-of-sentence token. Note that the LSTM reads the input sentence in reverse, because doing so introduces many short term dependencies in the data that make the optimization problem much easier.

Attention结构回顾



Soft Attention



Hard Attention

Attention结构回顾

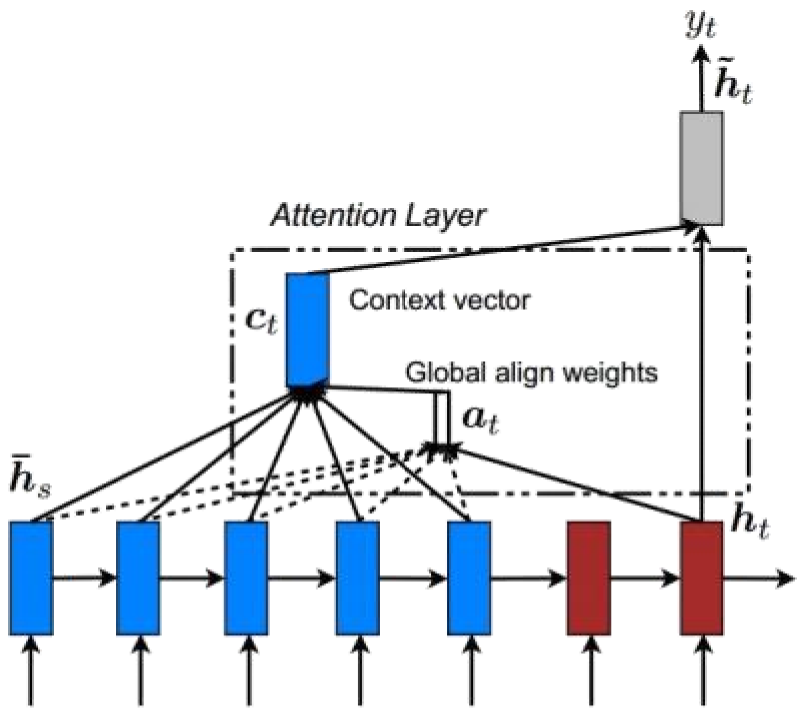


Figure 2: **Global attentional model** – at each time step t, the model infers a *variable-length* alignment weight vector $\boldsymbol{a}_t$ based on the current target state $\boldsymbol{h}_t$ and all source states $\bar{\boldsymbol{h}}_s$. A global context vector $\boldsymbol{c}_t$ is then computed as the weighted average, according to $\boldsymbol{a}_t$, over all the source states.

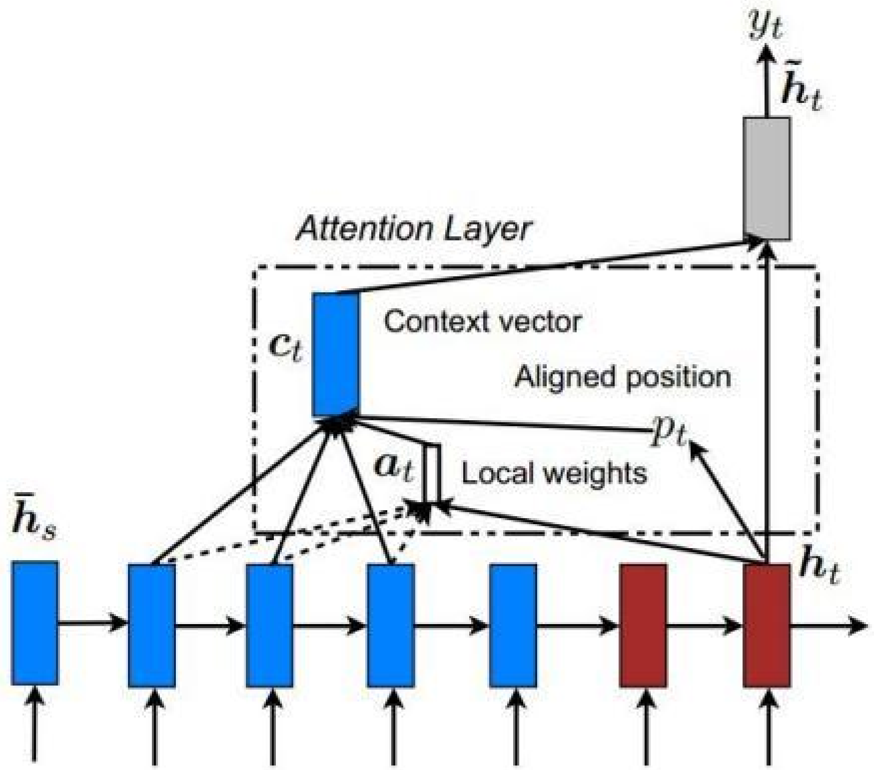
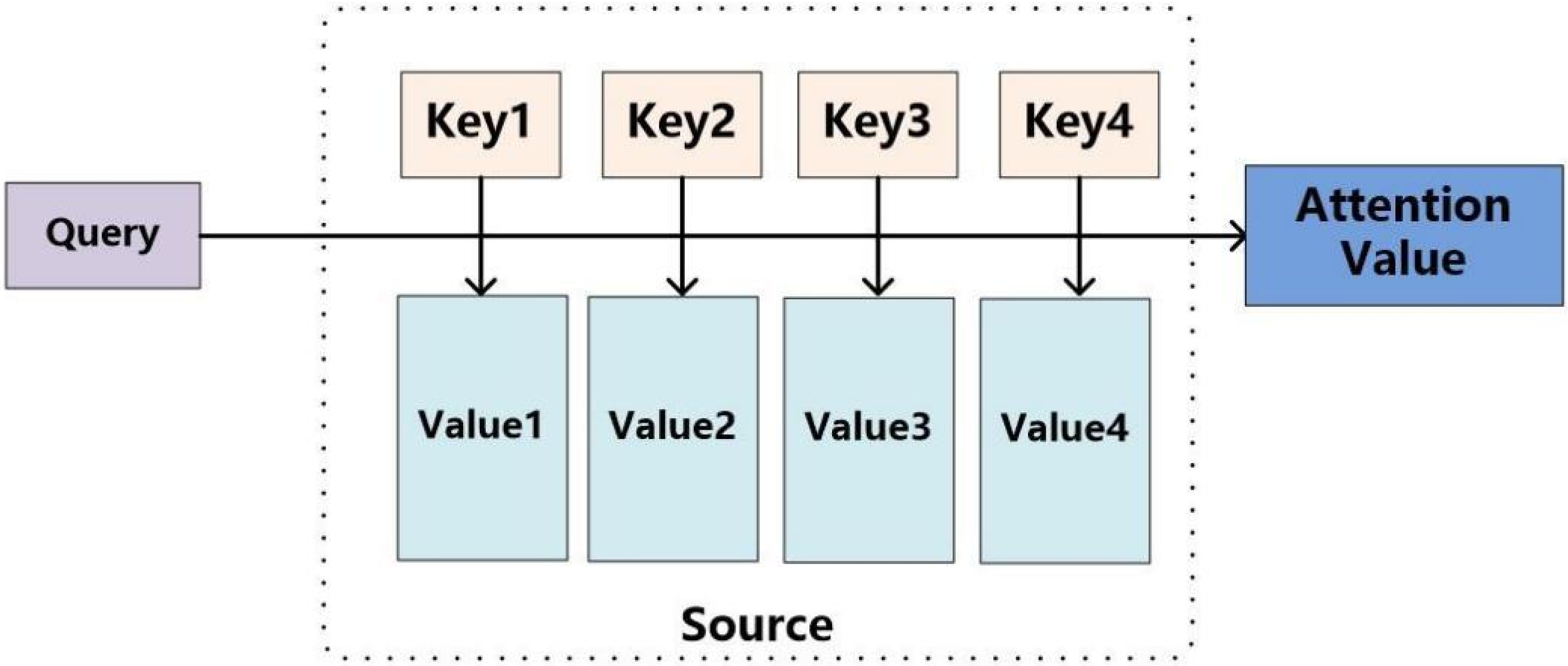


Figure 3: **Local attention model** – the model first predicts a single aligned position p_t for the current target word. A window centered around the source position p_t is then used to compute a context vector $\boldsymbol{c}_t$, a weighted average of the source hidden states in the window. The weights $\boldsymbol{a}_t$ are inferred from the current target state $\boldsymbol{h}_t$ and those source states $\bar{\boldsymbol{h}}_s$ in the window.

Attention结构回顾



Attention结构回顾

$$e_{t,i} = s_{t-1}^{T} h_i$$

$$e_{t,i} = s_{t-1}^{T} h_i / \sqrt{d}$$

$$e_{t,i} = s_{t-1}^{T} W h_i$$

$$e_{t,i} = u^{T} \tanh\left(W_1 h_i + W_2 s_{t-1}\right)$$

$$e_{t,i} = W_1 h_i + W_2 s_{t-1}$$

$$e_{t,i} = W h_i$$

Self Attention

- 在17年被提出于《Attention Is All You Need, Ashish Vaswani》，也称为Transformer结构；内部包含Multi-Head Attention以及Rest残差结构。
- Transformer是Bert网络结构的基础。
- https://arxiv.org/pdf/1706.03762.pdf

https://colab.research.google.com/github/tensorflow/tensor2tensor/blob/master/tensor2tensor/notebooks/hello_t2t.ipynb#scrollTo=OJKU36QAfqOC

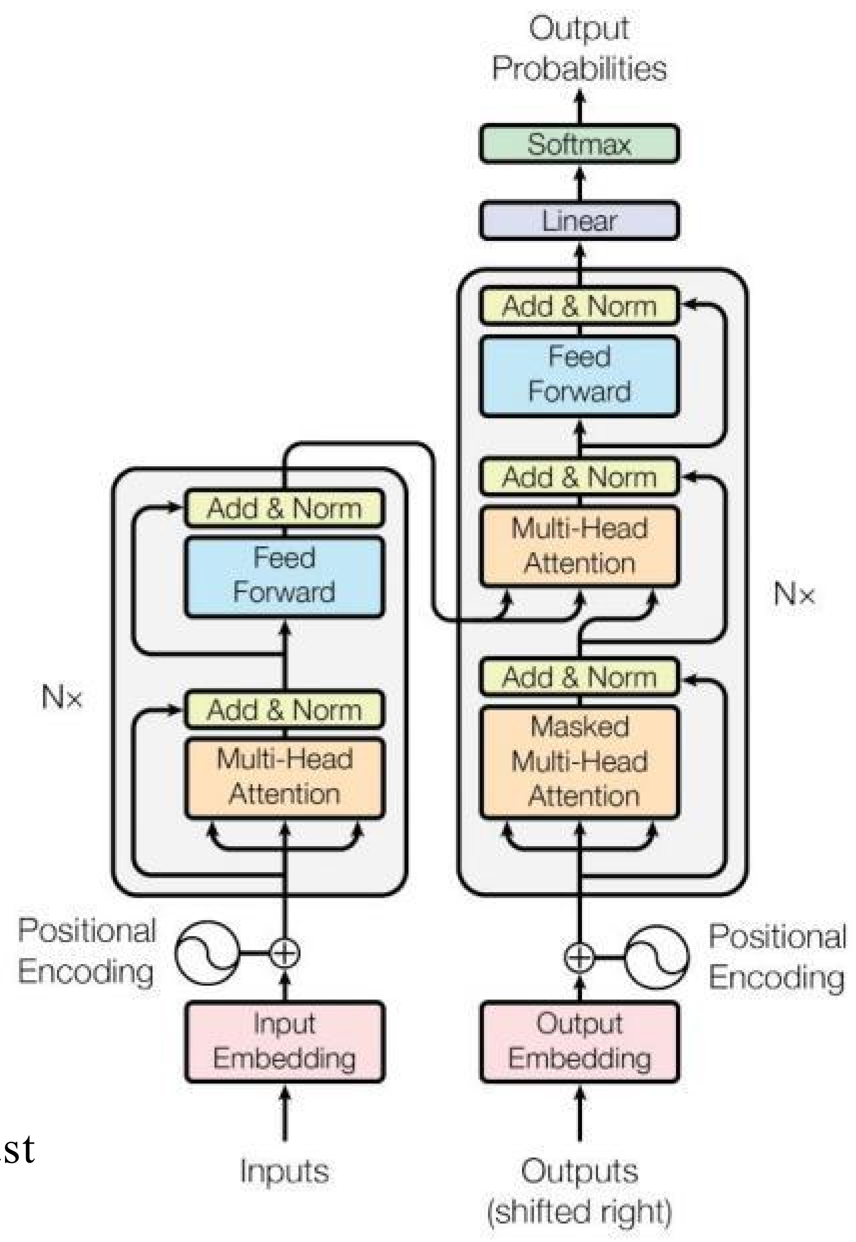


Figure 1: The Transformer - model architecture.

Transformer

- 传统缺点：seq2seq使用循环网络固有的顺序特性阻碍样本训练的并行化，这在更长的序列长度上变得至关重要，因为有限的内存限制样本的批次大小。
- 新结构：Transformer，这种模型架构避免循环并完全依赖于attention机制来绘制输入和输出之间的全局依赖关系。Transformer允许进行更多的并行化。
- Self-attention：有时称为intra-attention，是一种attention机制，它关联单个序列的不同位置以计算序列的表示。Self-attention已成功用于各种任务，包括阅读理解、摘要概括、文本蕴涵和学习与任务无关的句子表征。

Transformer

- Transformer: Attention Is All You Need
  - 2017, Google, https://arxiv.org/pdf/1706.03762.pdf
  - New Features:
    - **Self-Attention：核心-提取特征信息；**
    - **Multi-Head-Attention：相当于定义多个self-attention提取不同方面的特征信息；**
    - **Positional Encoding：通过增加位置embedding，提高模型的 序列特征提取能力；**
    - **Residuals：残差模块，防止模型退化；**
    - **Layer Norm：正则化模块，加快训练速度，防止模型过拟合；**
    - **FFN：两层全连接**
    - **Masked：解码器中的Masked Multi Head Self Attention**

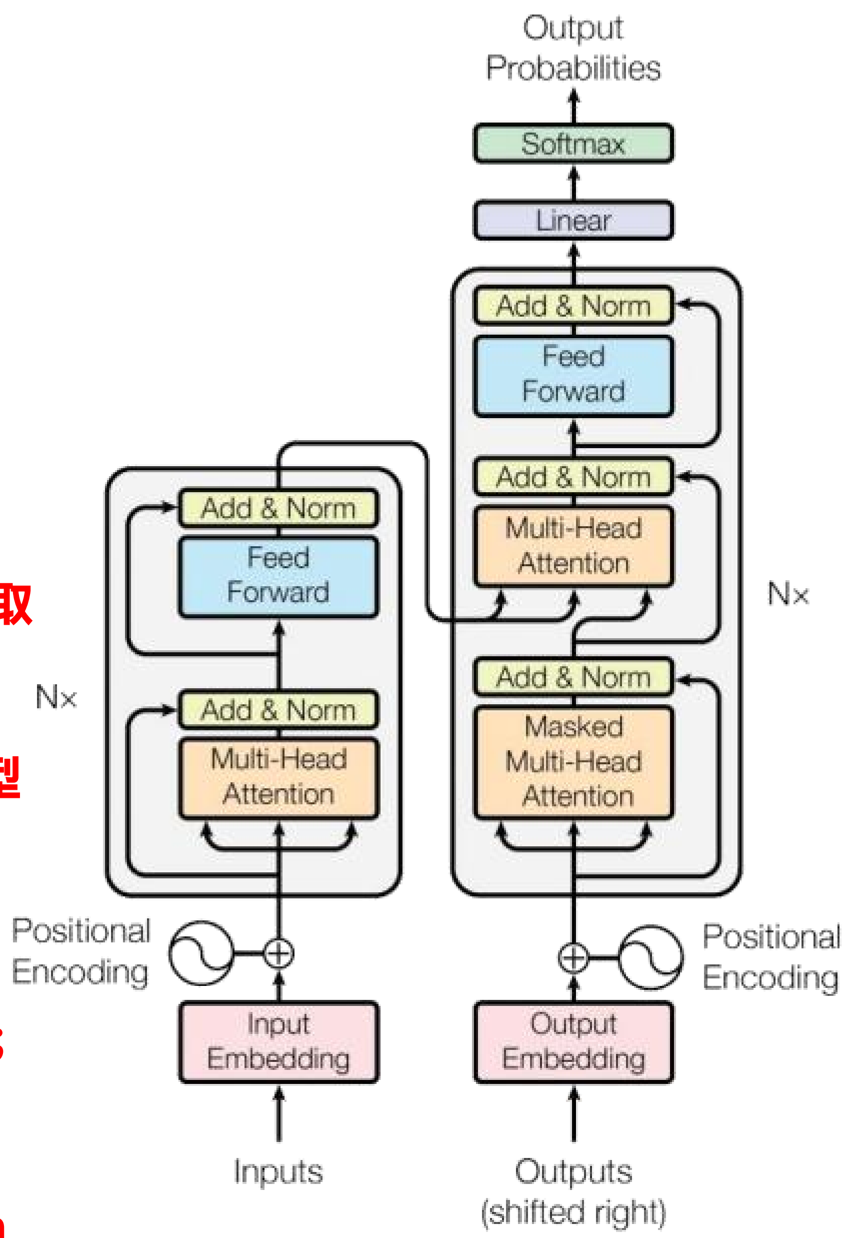
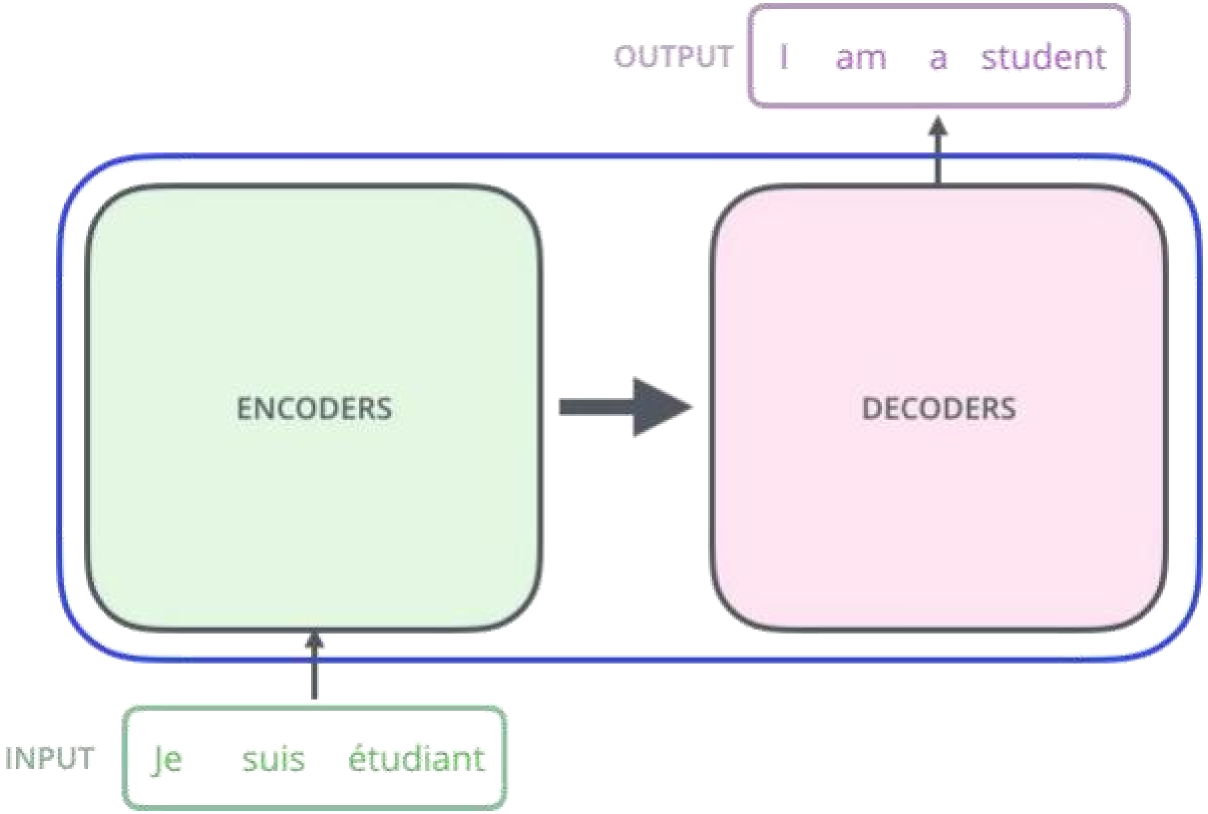


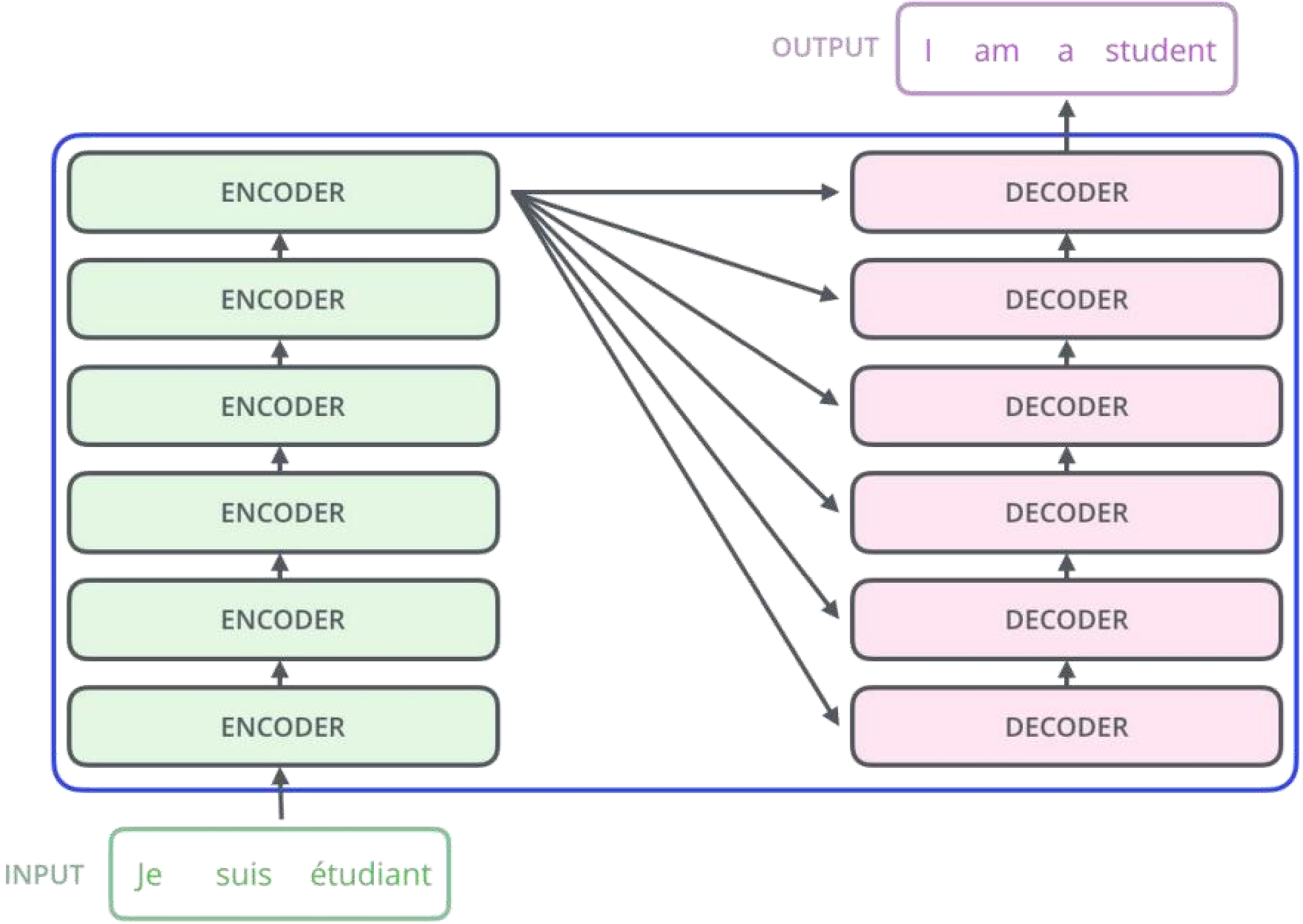
Figure 1: The Transformer - model architecture.

Transformer

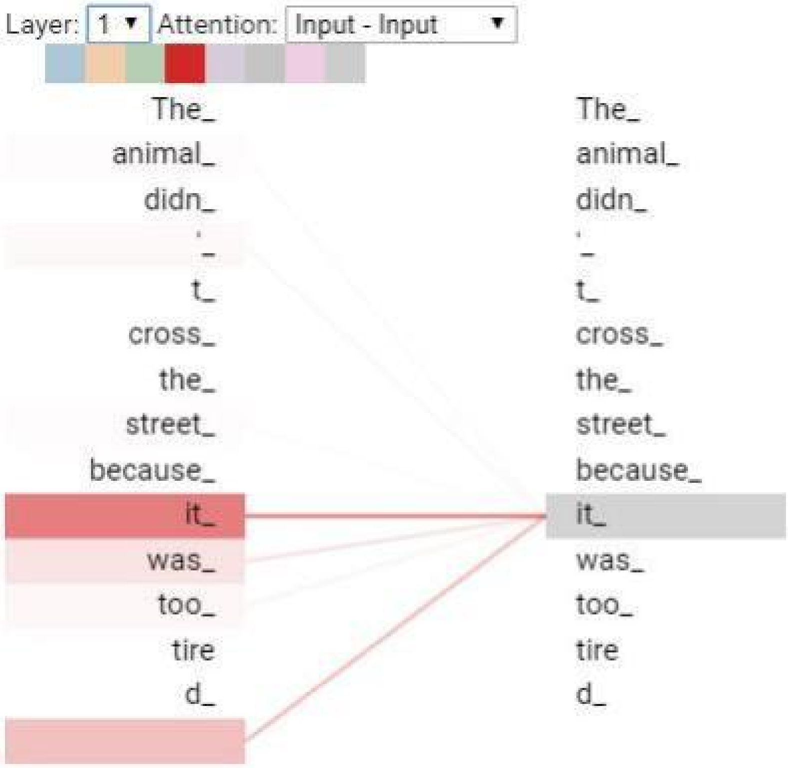


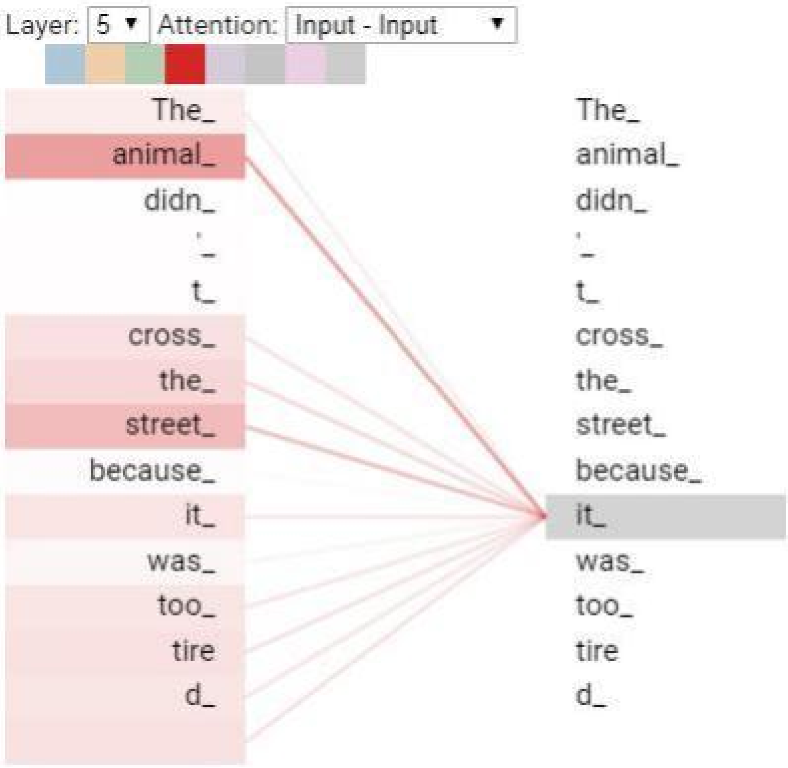
Transformer

六层Encoder和Decoder结构；各个Encoder和Decoder之间不共享权重Weights；



Transformer

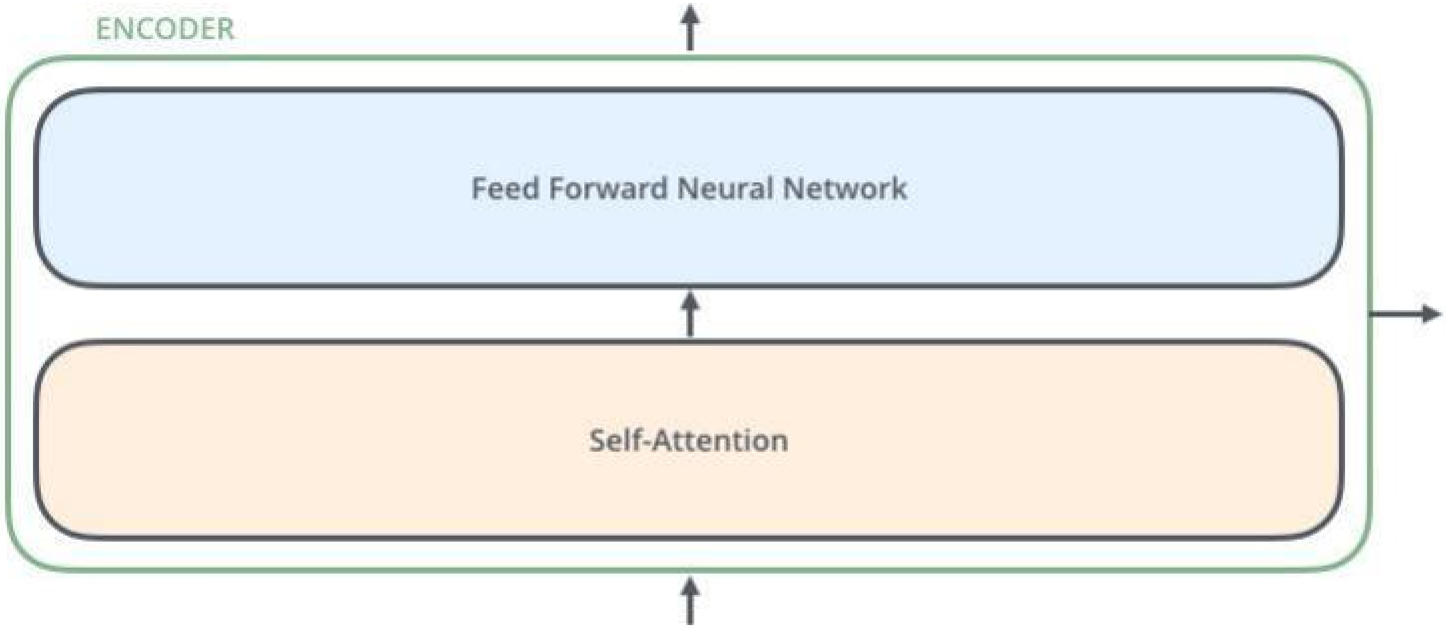




The animal didn't cross the street because it was too tired

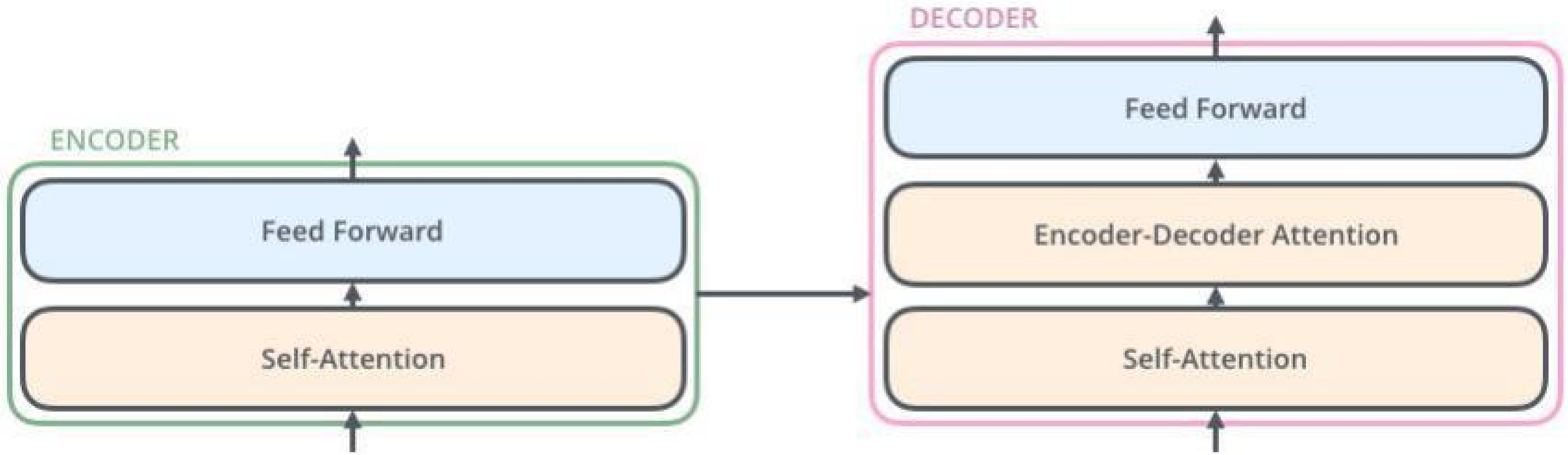
Transformer

每个Encoder包含两层结构: Self-Attention以及feed-forward NN构成。

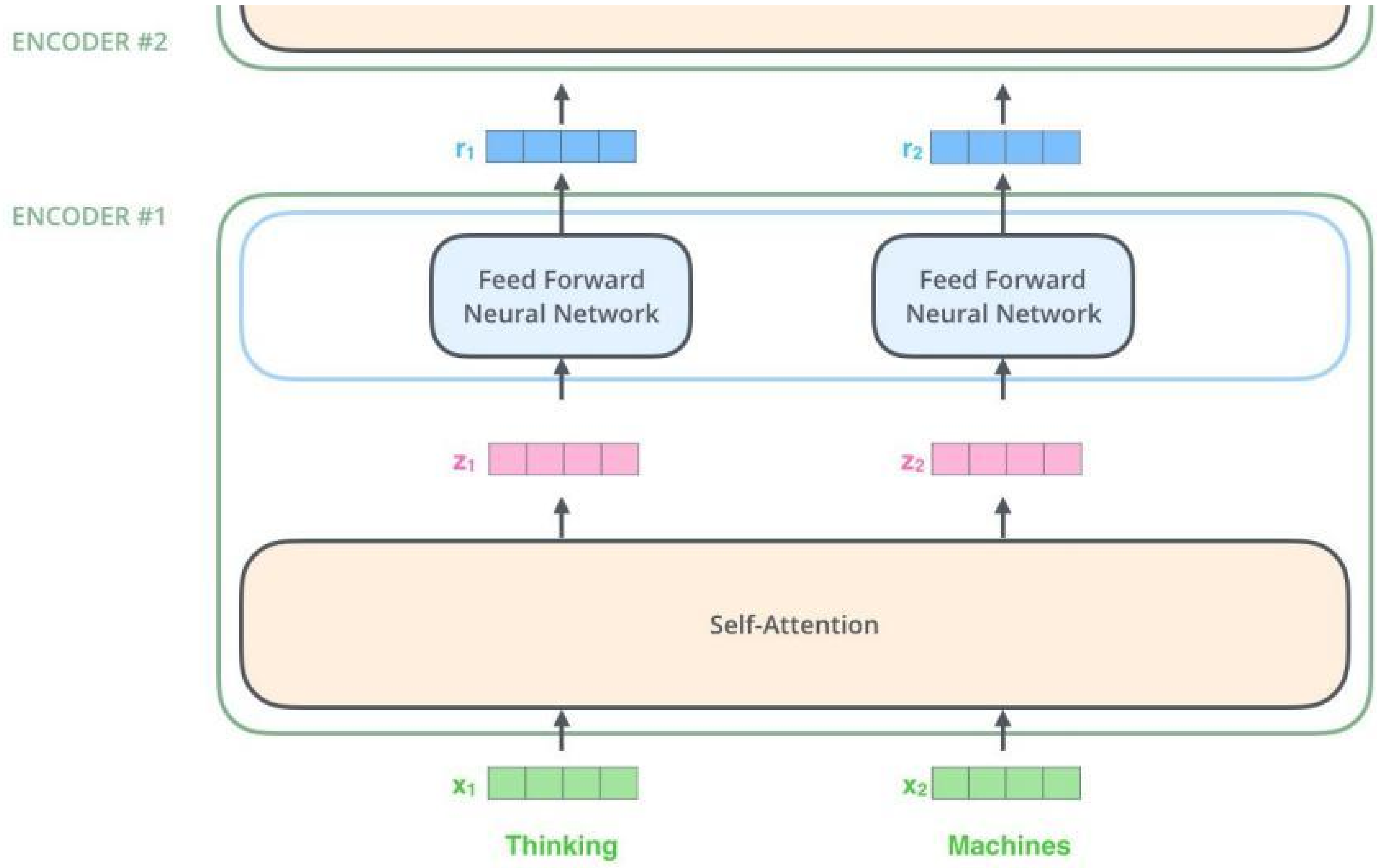


Transformer

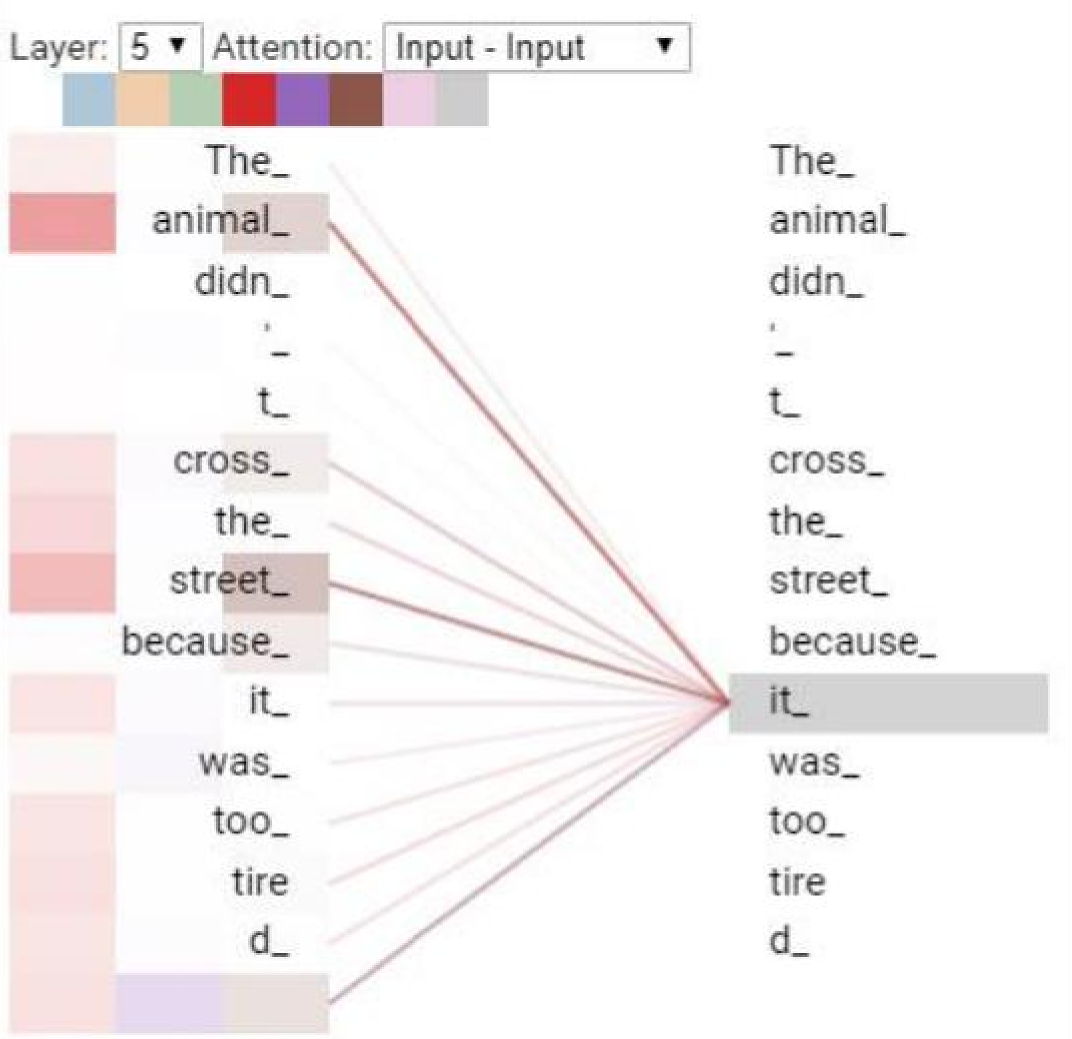
每个Decoder在Encoder的基础上，在Self-Attention和feed-forward NN之间增加了一个Encoder-Decoder Attention



Transformer
Encoder



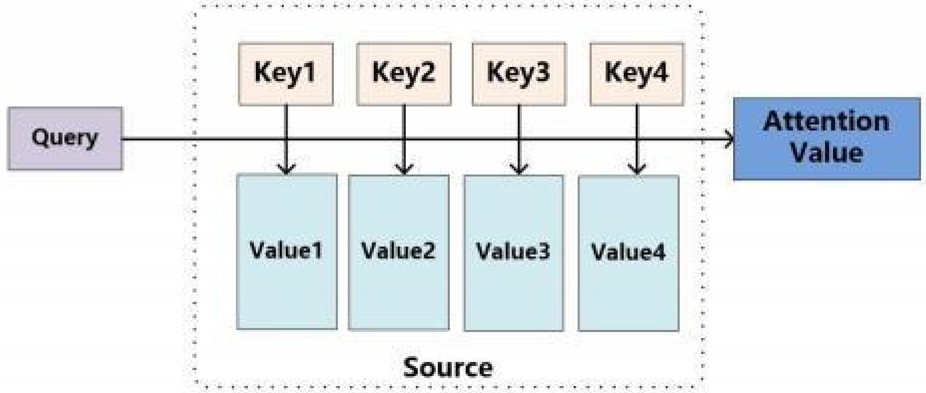
Transformer
Encoder Self-Attention

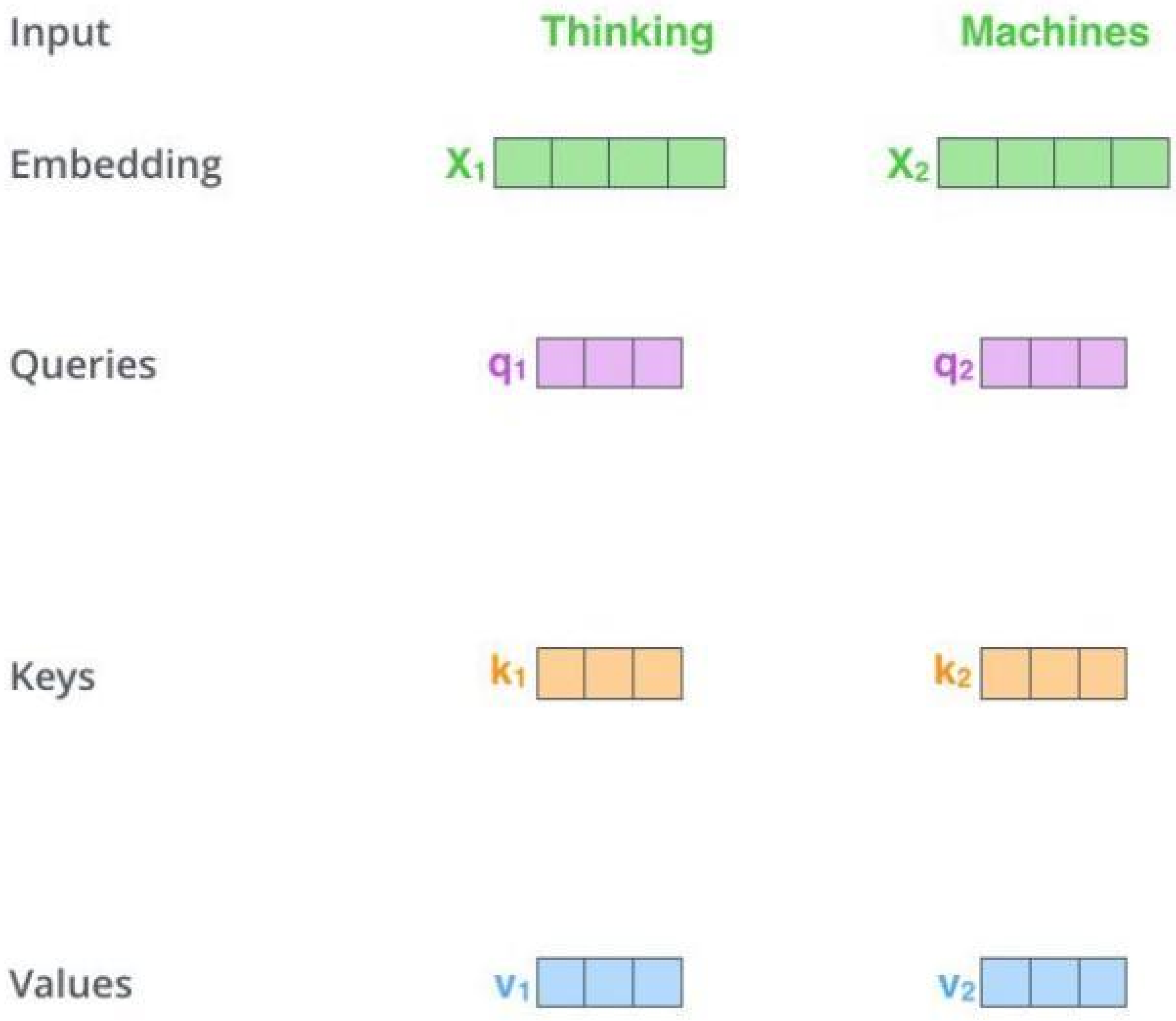


The animal didn't cross the street because it was too tired

Transformer

Encoder Self-Attention





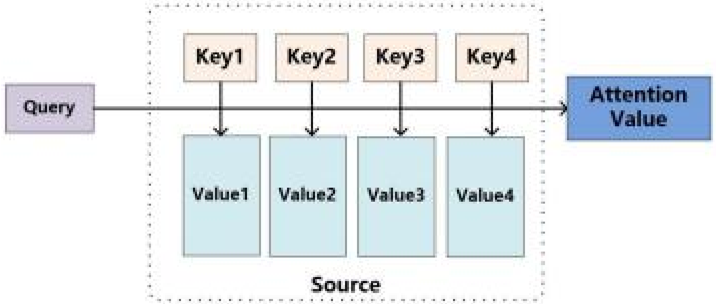
$$q = XW^Q$$

$$k = XW^K$$

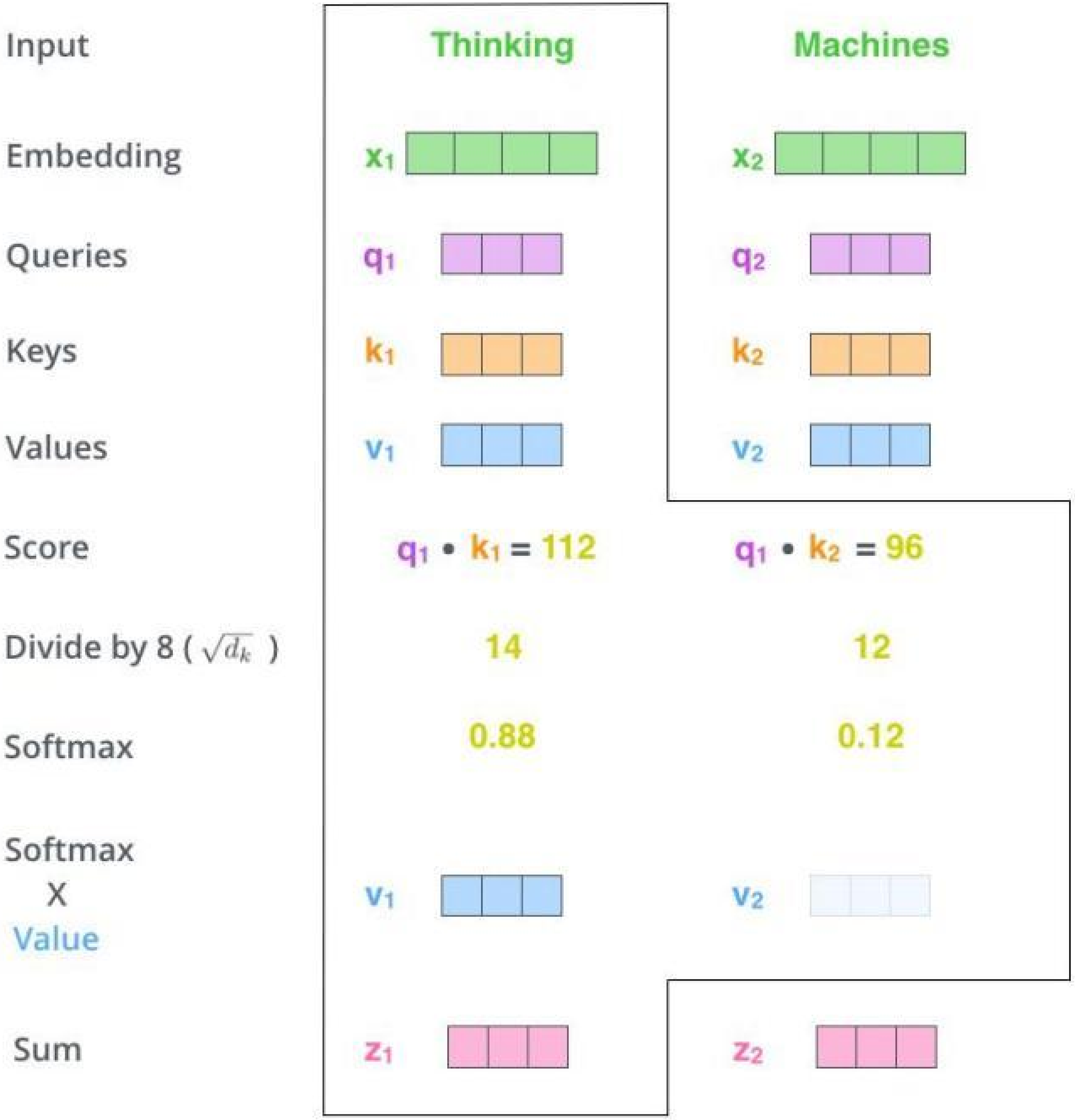
$$v = XW^V$$

Transformer

Encoder Self-Attention

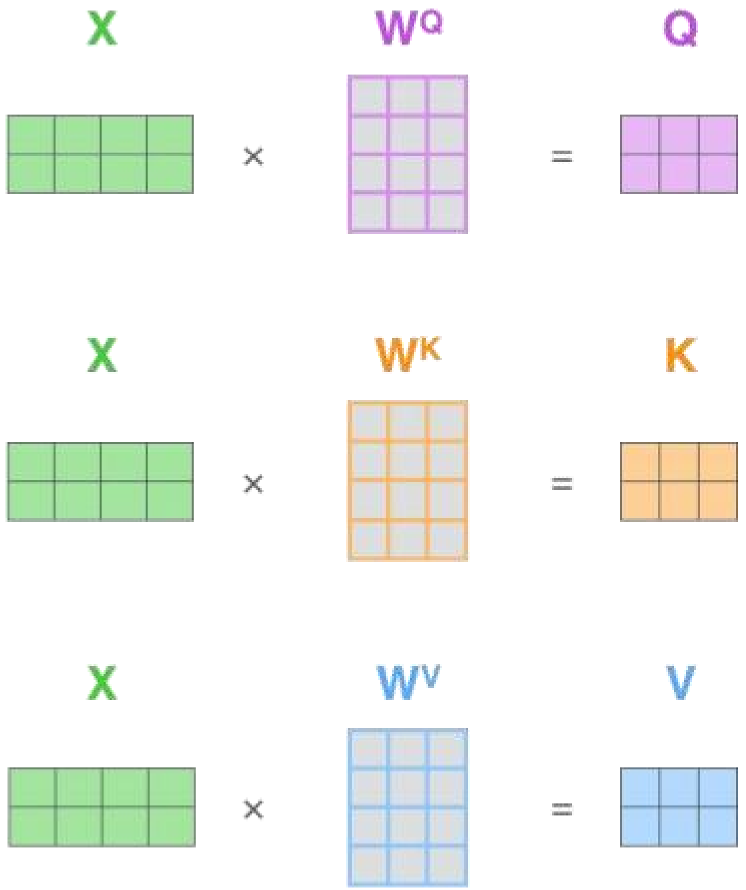


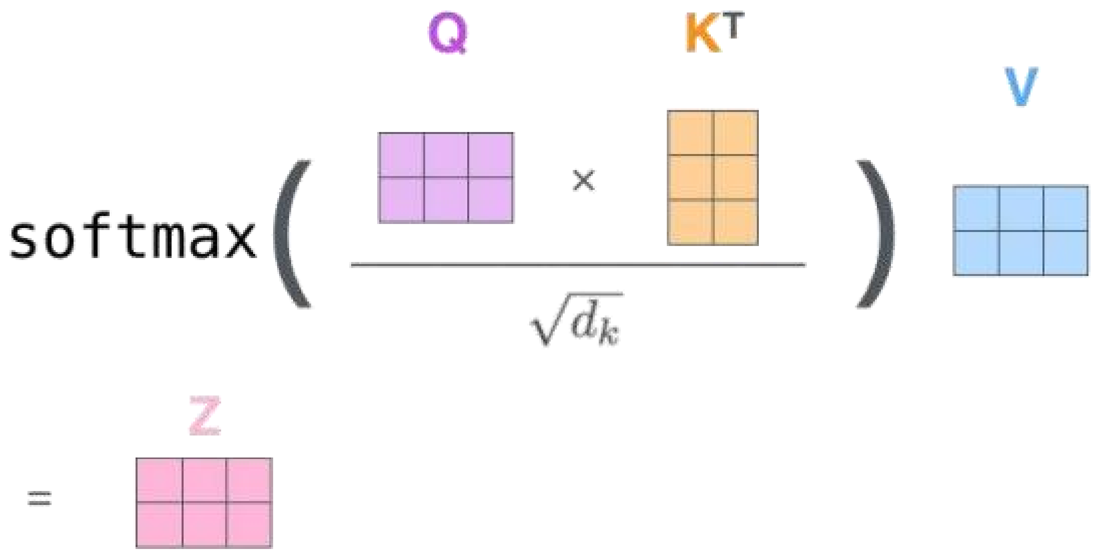
$$e_{t,i} = s_{t-1}^{T} h_i / \sqrt{d}$$



Transformer

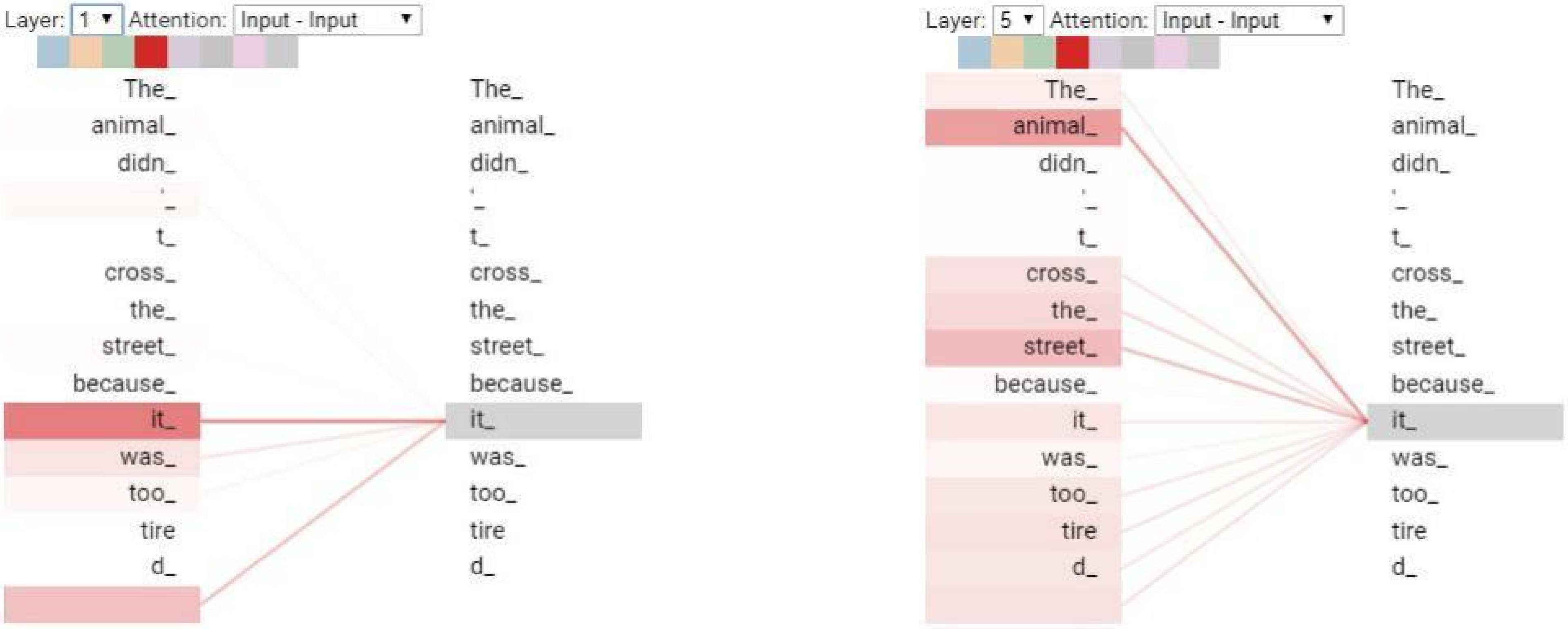
Encoder Self-Attention





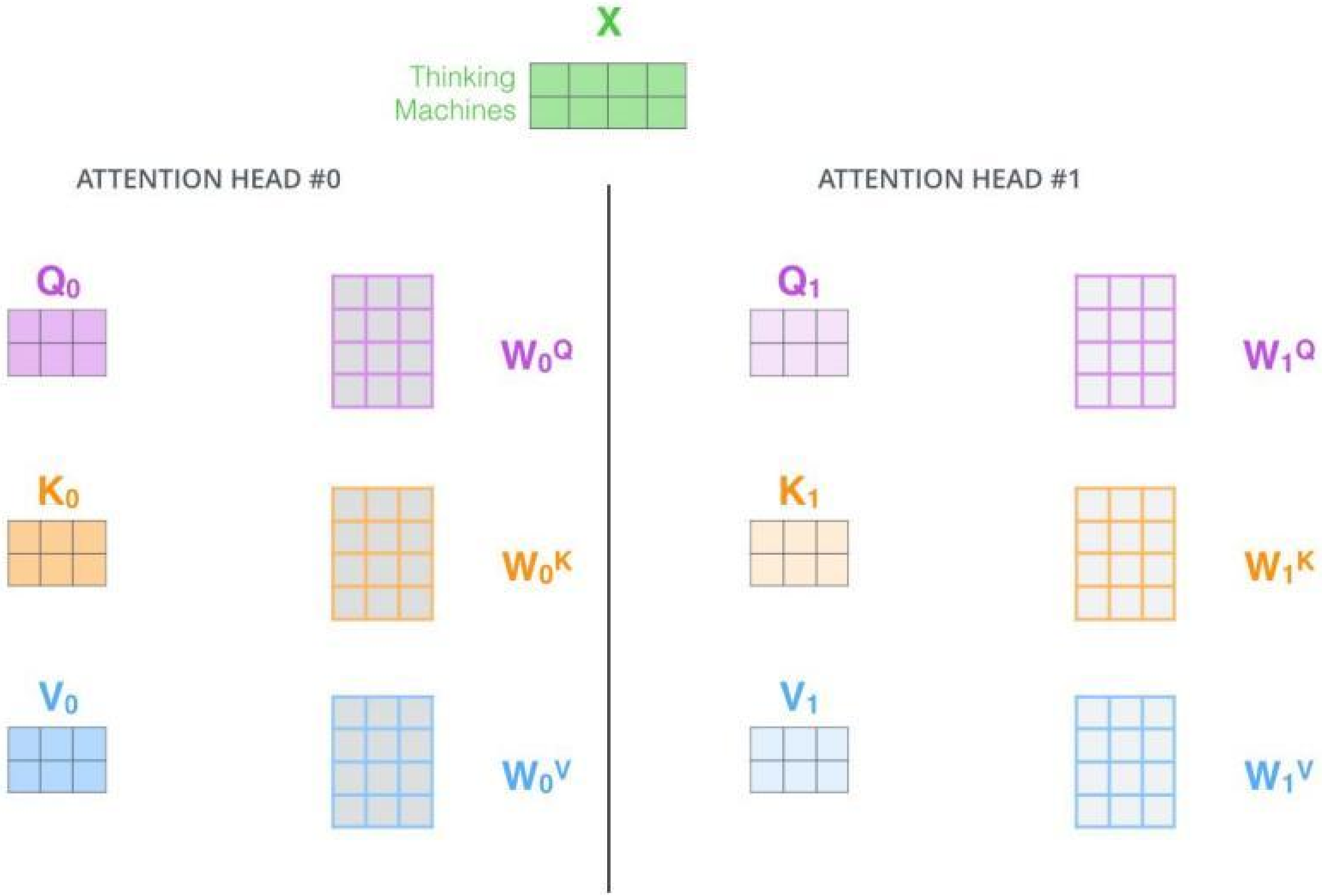
Transformer

Encoder Self-Attention



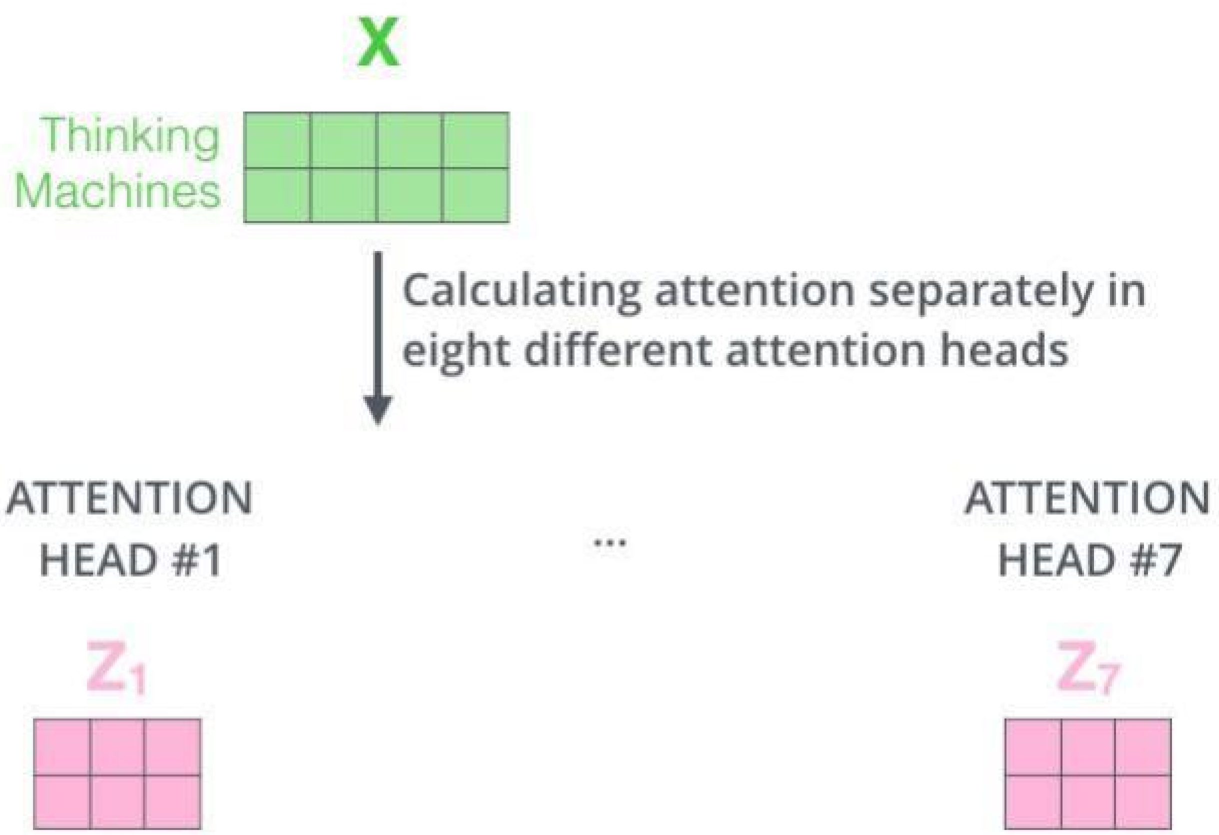
Transformer

Encoder Multi-Headed-Attention



Transformer

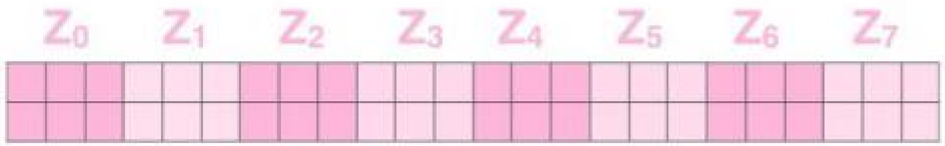
Encoder Multi-Headed-Attention



Transformer

Encoder Multi-Headed-Attention

1) Concatenate all the attention heads

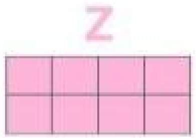


2) Multiply with a weight matrix W^O that was trained jointly with the model

X

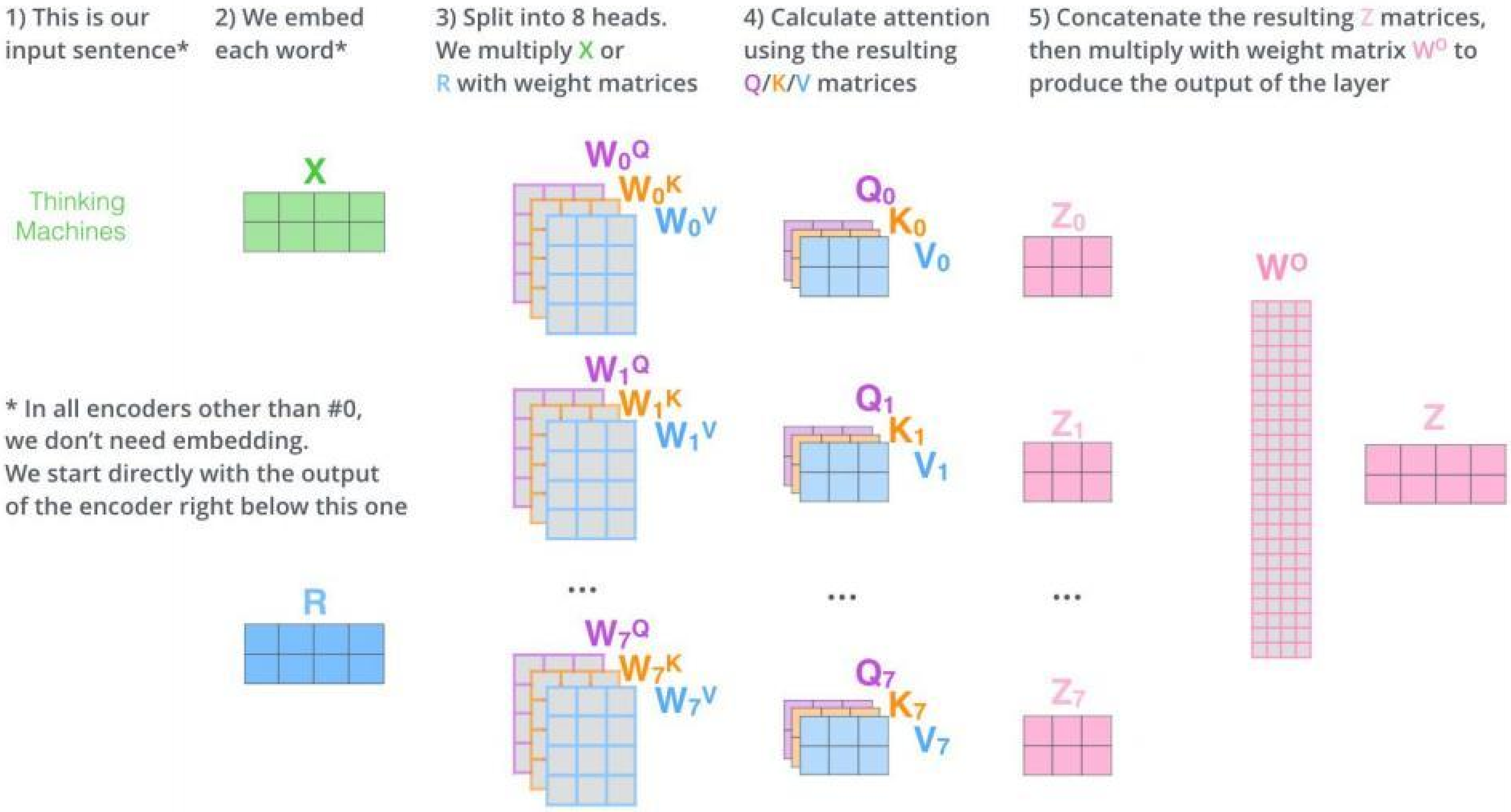
3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN

=



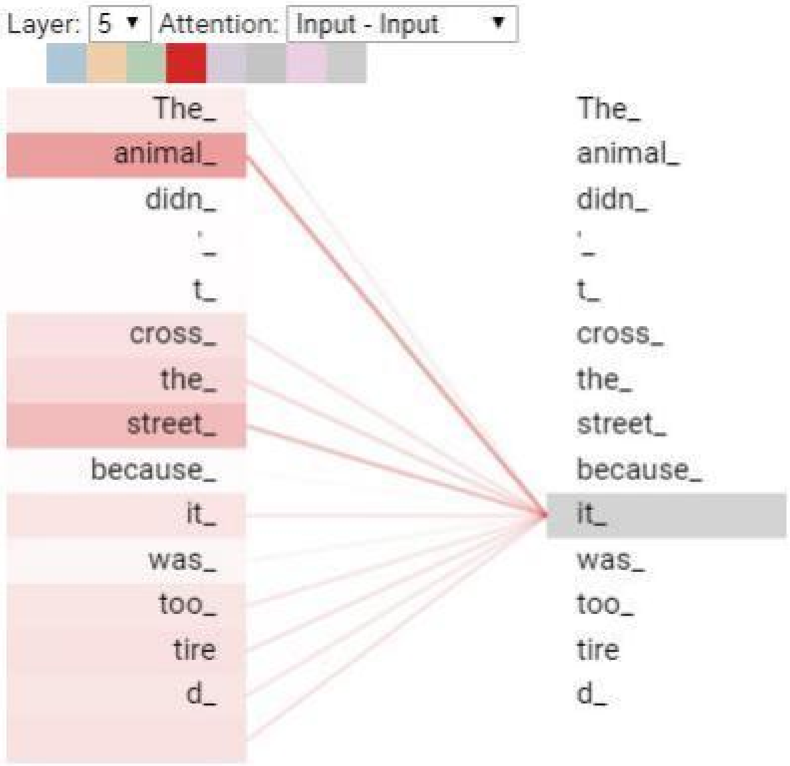
Transformer

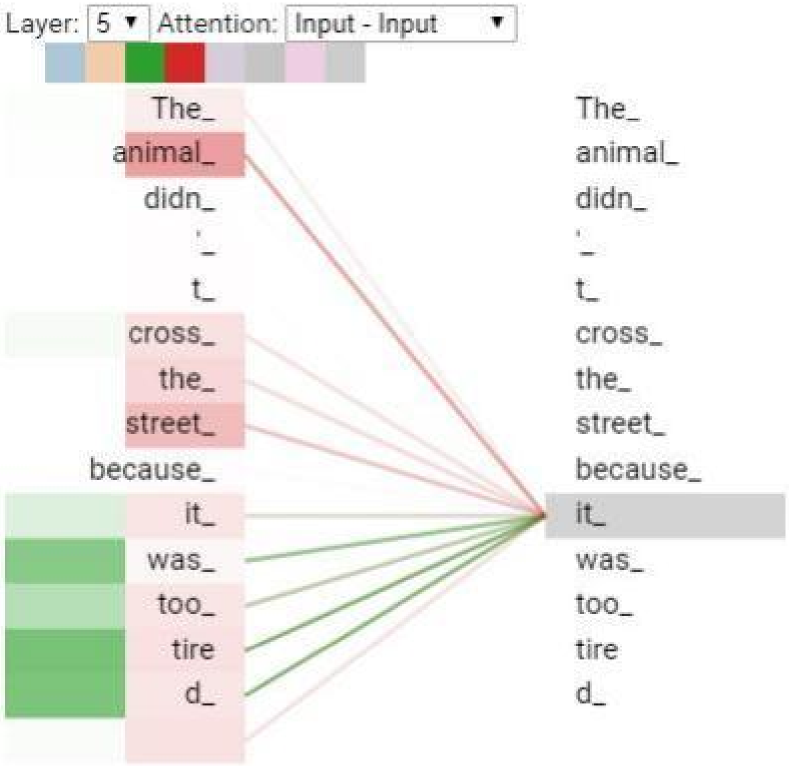
Encoder Multi-Headed-Attention



Transformer

Encoder Multi-Headed-Attention





Transformer

Positional Encoding

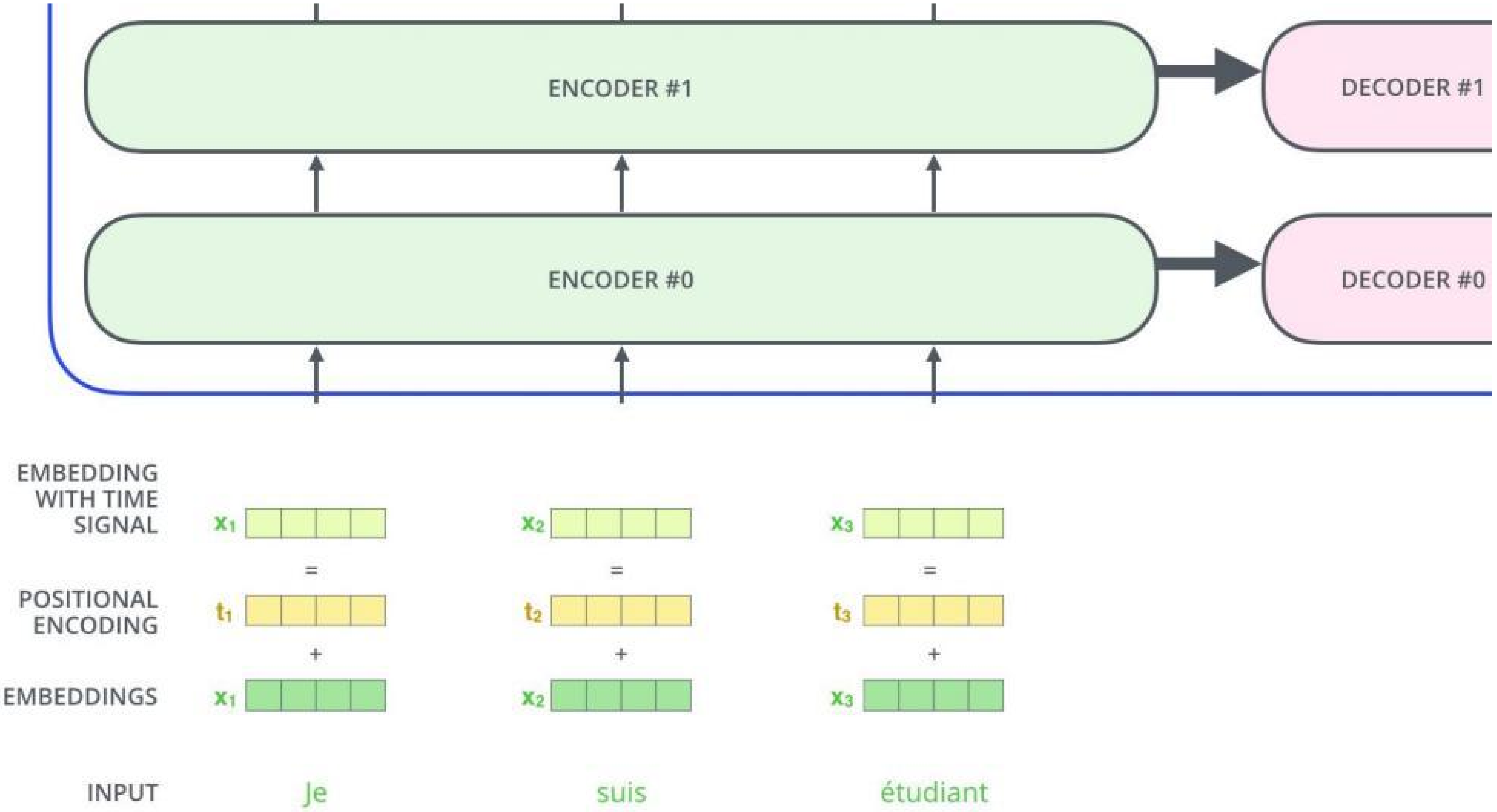
- 模型还没有描述词之间的顺序关系，也就是如果将一个句子打乱其中的位置，也应该获得相同的注意力，为了解决这个问题，论文加入了自定义位置编码，位置编码和word embedding长度相同的特征向量，然后和word embedding进行求和操作。

$$PE(pos, 2i) = sin(\frac{pos}{10000^{\frac{2i}{d_{model}}}})$$

$$PE(pos, 2i+1) = cos(\frac{pos}{10000^{\frac{2i}{d_{model}}}})$$

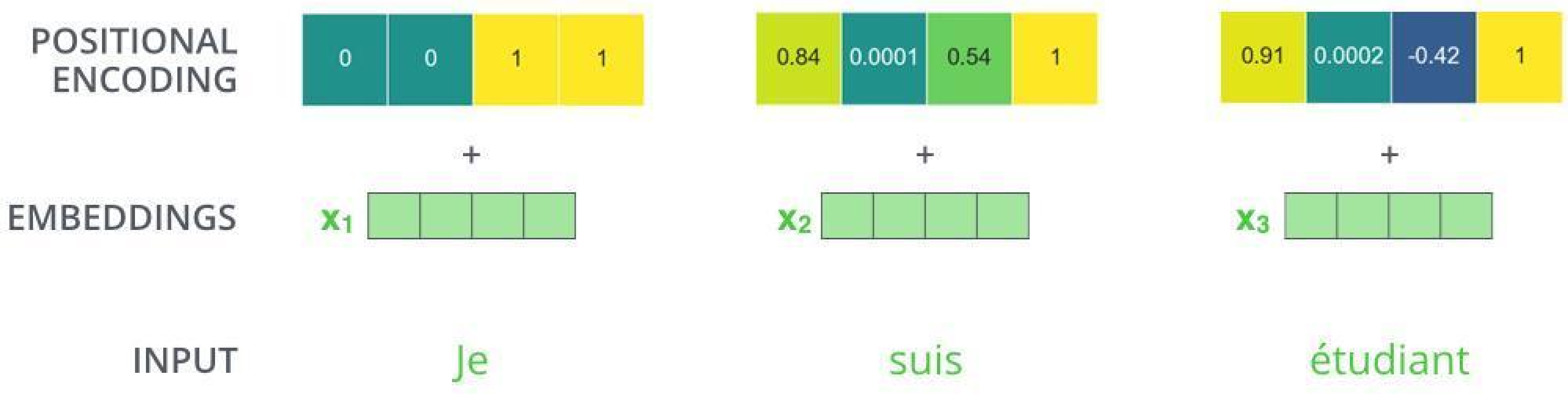
Transformer

Positional Encoding



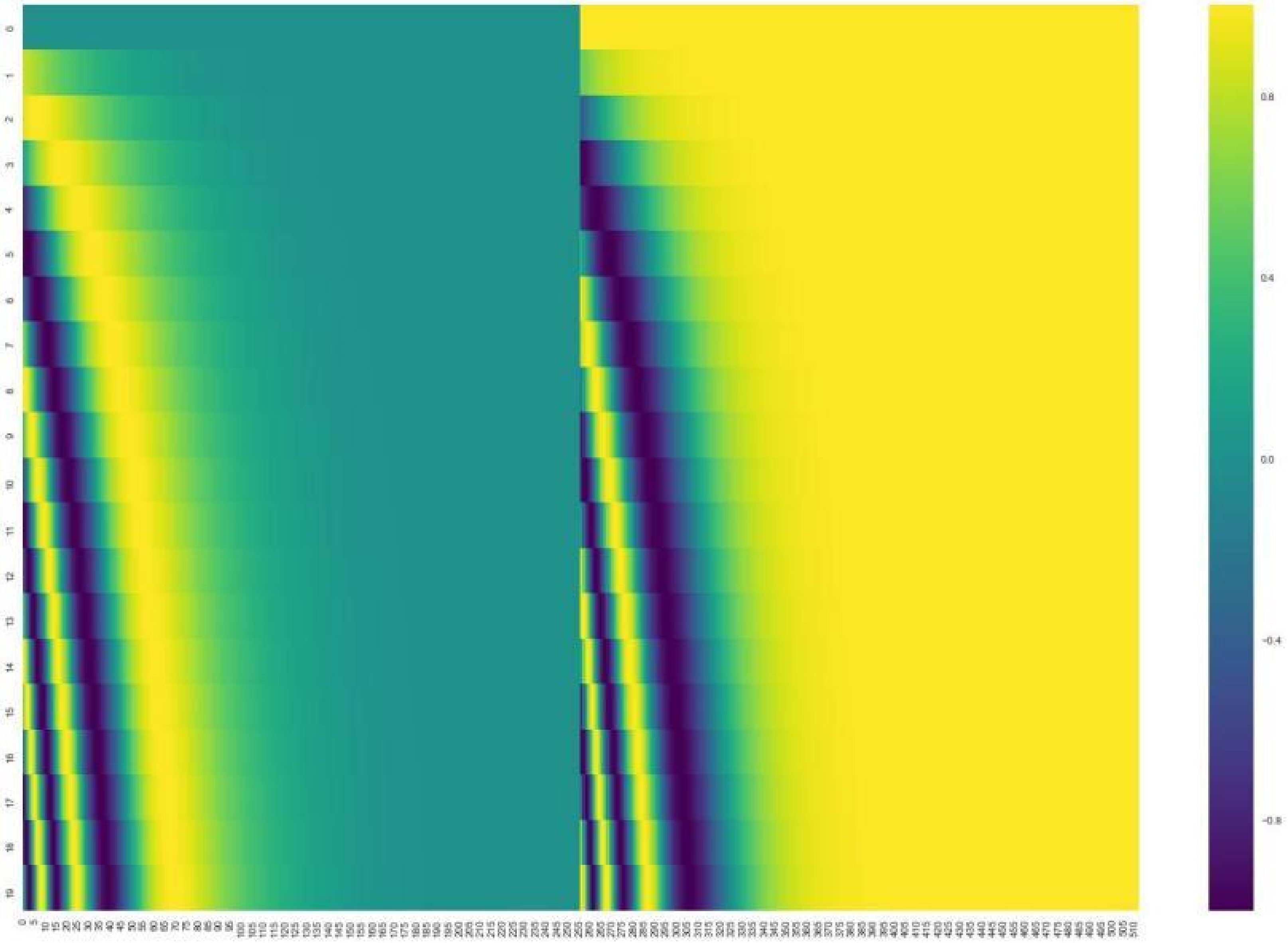
Transformer

Positional Encoding



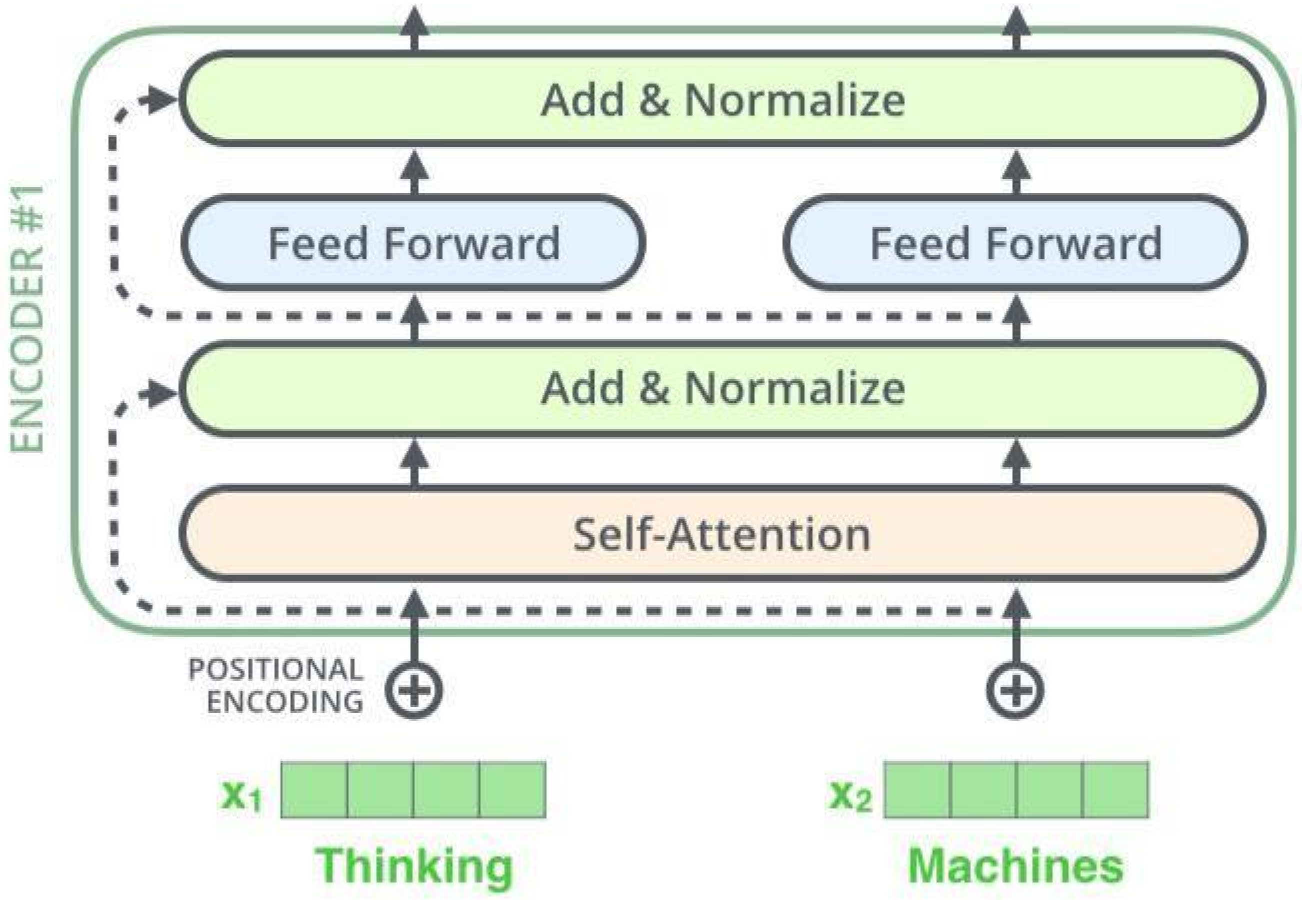
Transformer

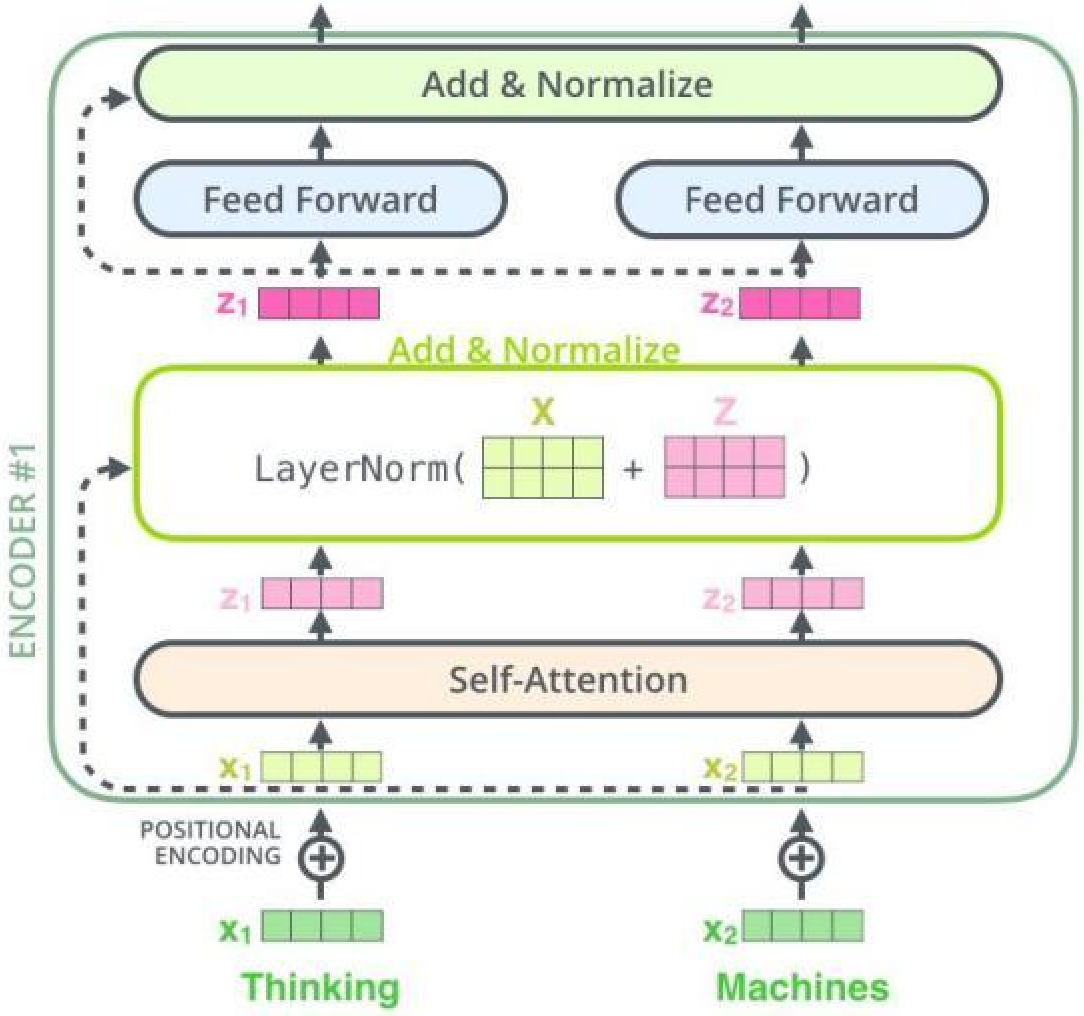
Positional Encoding



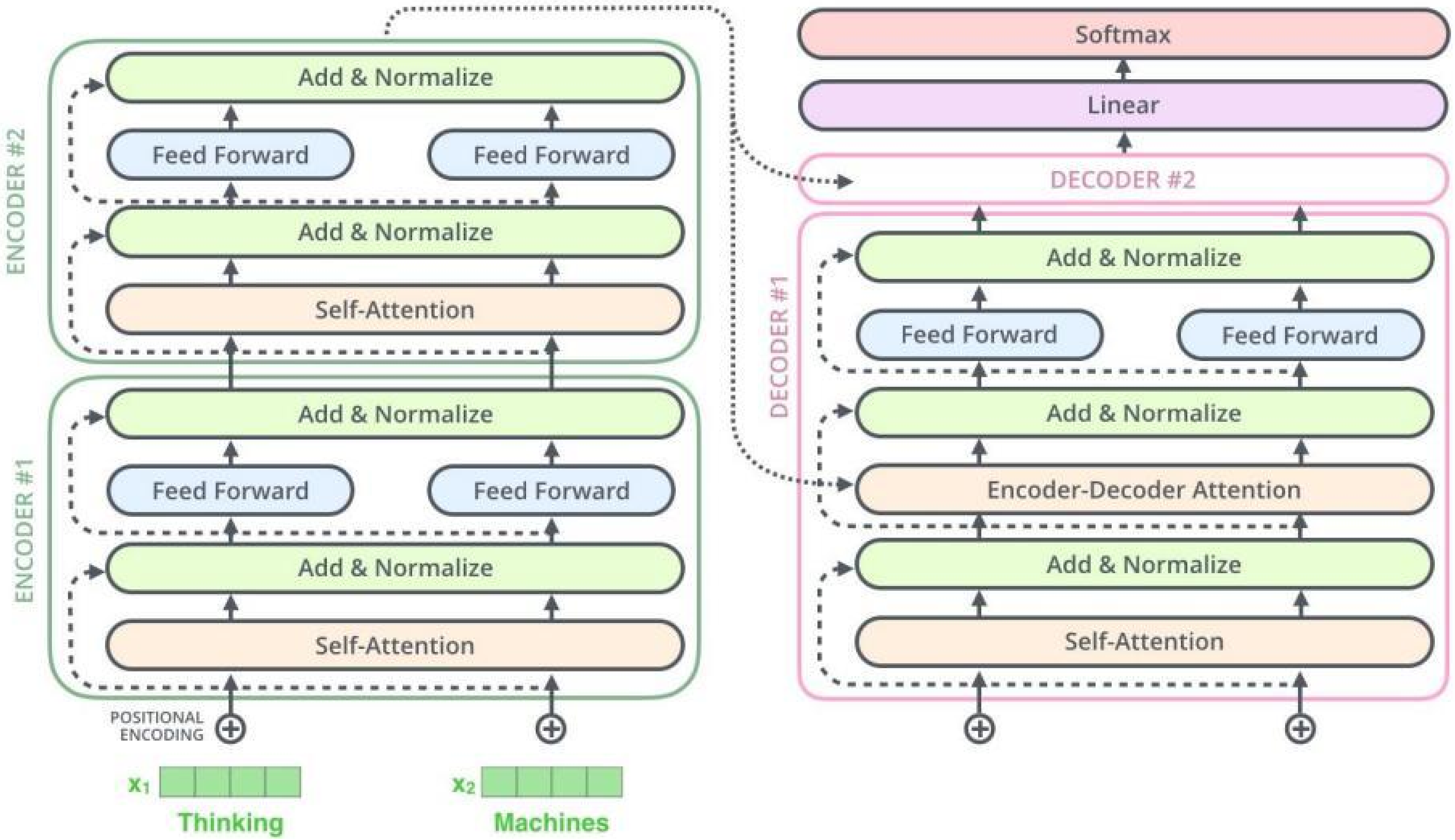
Transformer

LayerNorm&Residuals





Transformer
Decoder



Transformer
Decoder Masked Multi-Headed-Attention

- 和编码部分的multi-head attention类似， 但是多了一 次masked， 因为在解码部分， 解码的时候时从左到右依次解码的， 当解出第一个字的时候， 第一个字只能与第一个字计算相关性， 当解出第二个字的时候， 只能计算出第二个字与第一个字和第二个字的相关性。；

- 因此这里引入Mask的概念进行掩码操作。

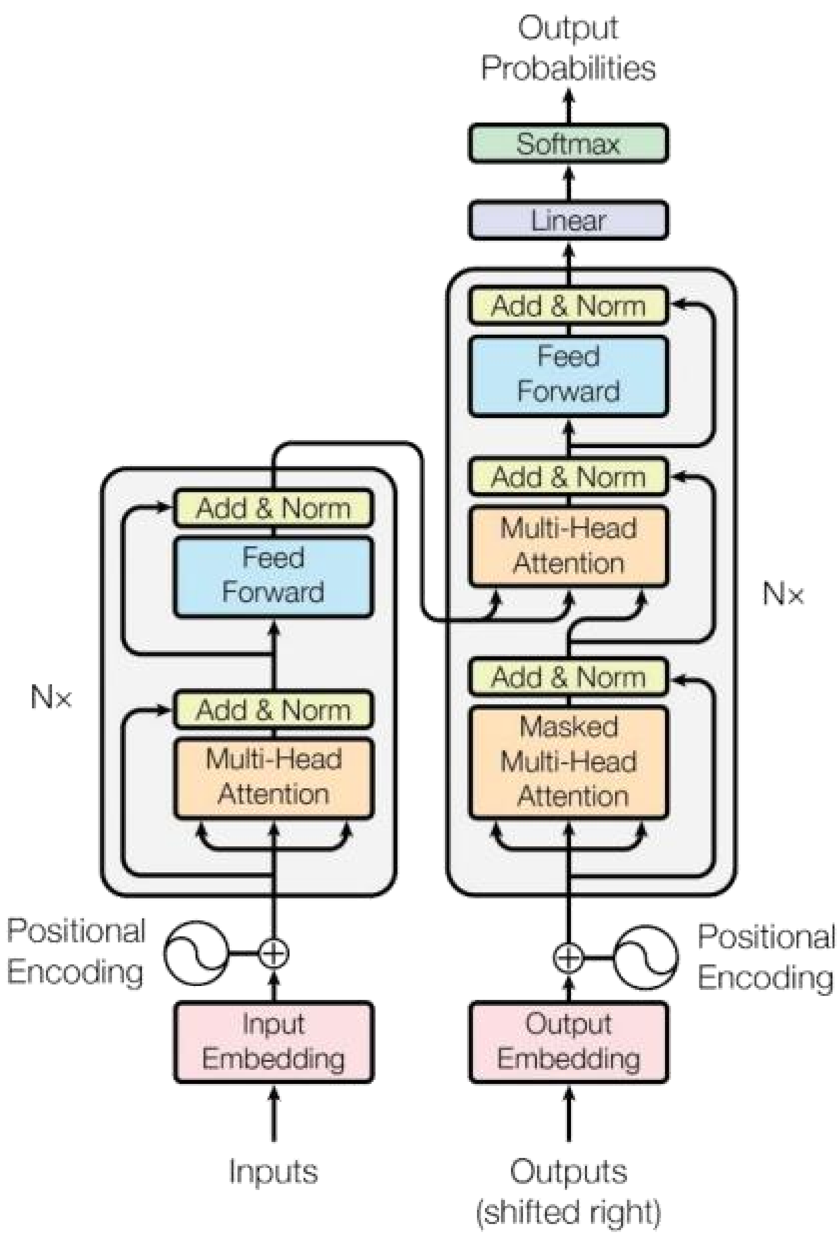
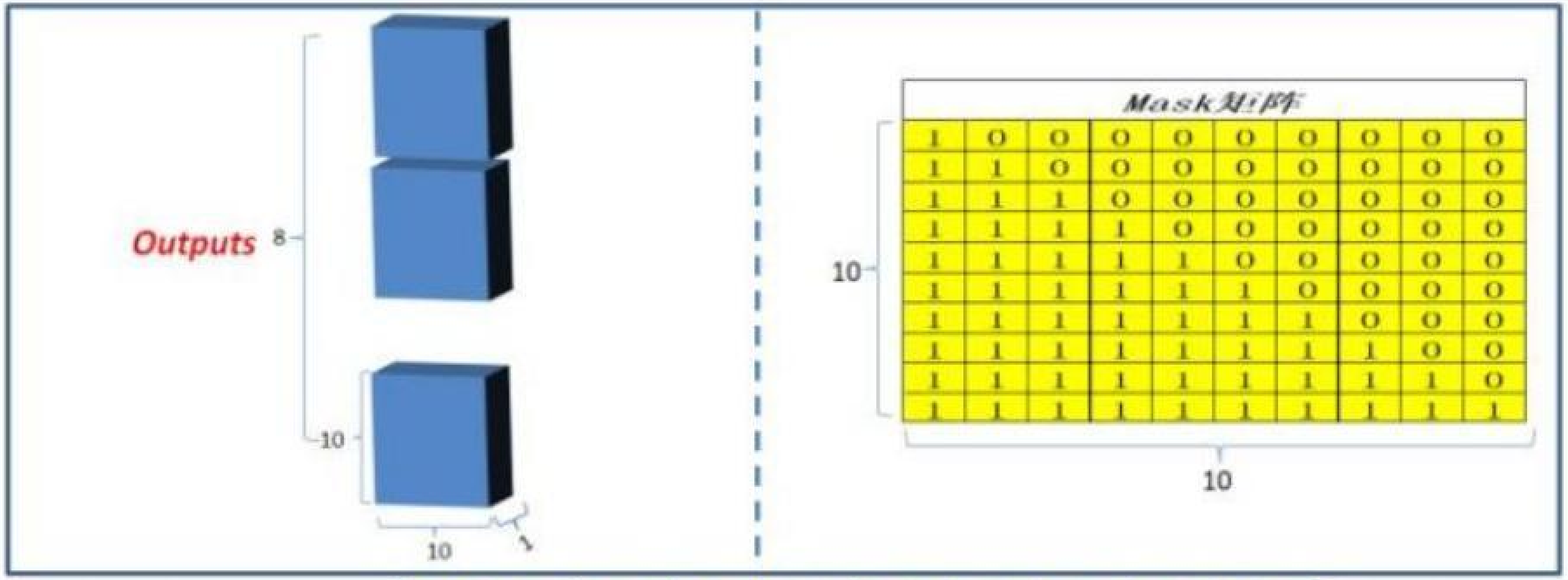
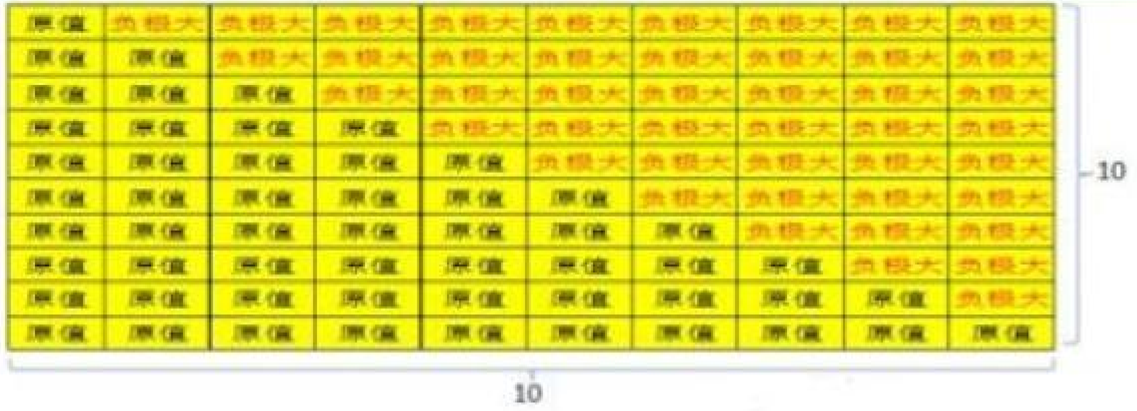


Figure 1: The Transformer - model architecture.

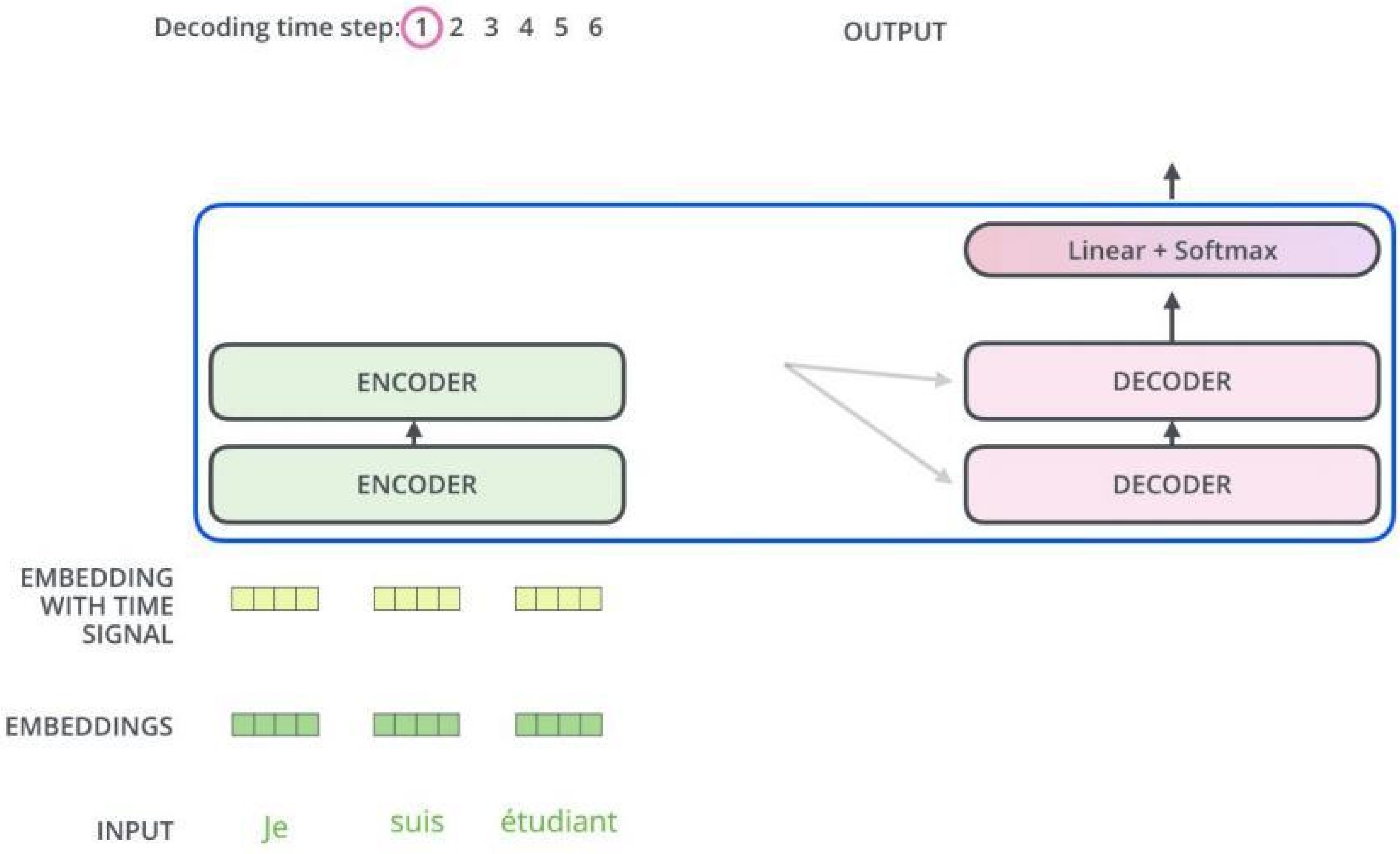
Transformer
Decoder Masked Multi-Headed-Attention



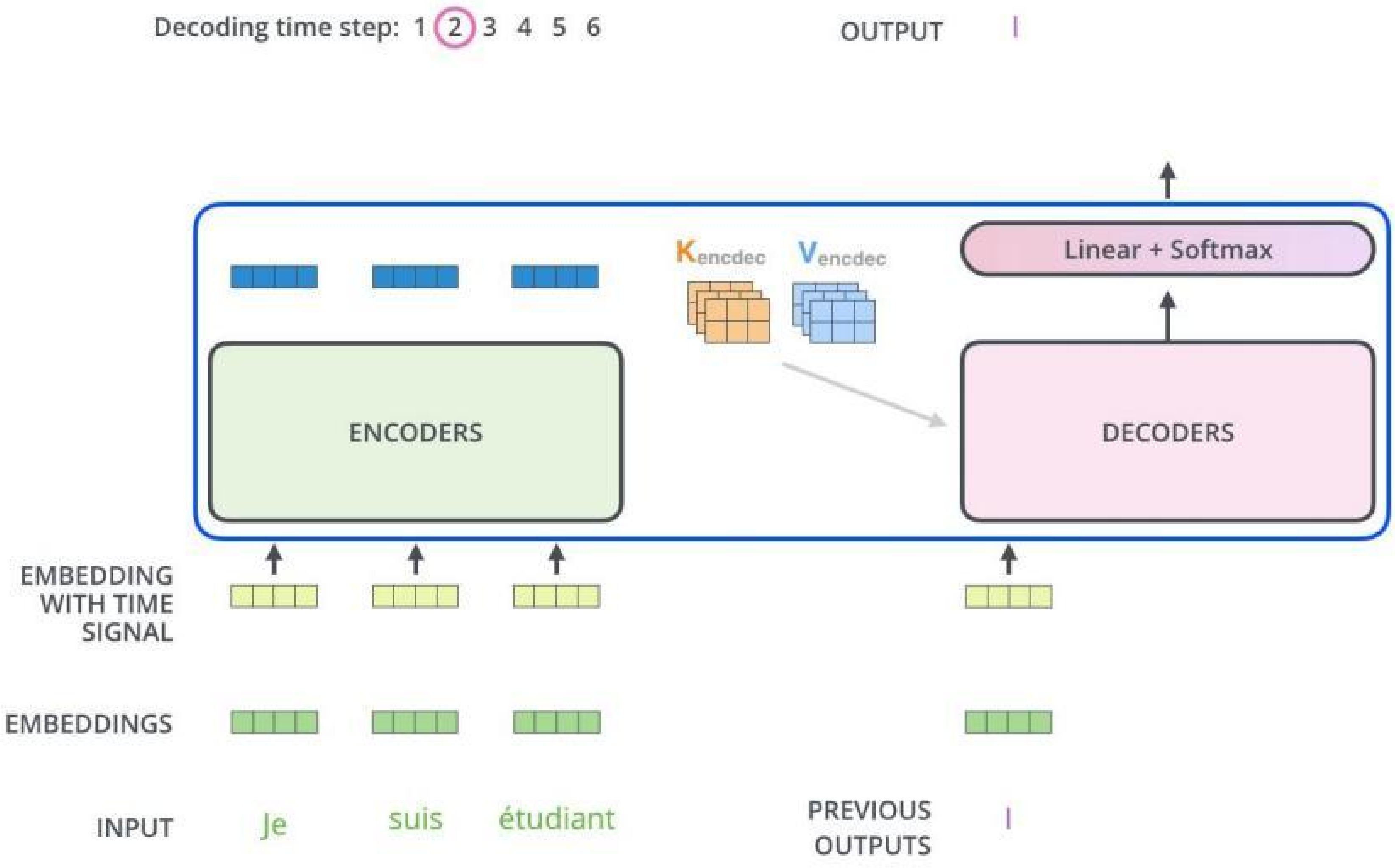
用Mask矩阵作用于Outputs中的每一个10X10单元矩阵：mask矩阵中元素‘1’对应的Outputs单元元素保留原值，‘0’对应的Outputs单元元素替换为负极大值；



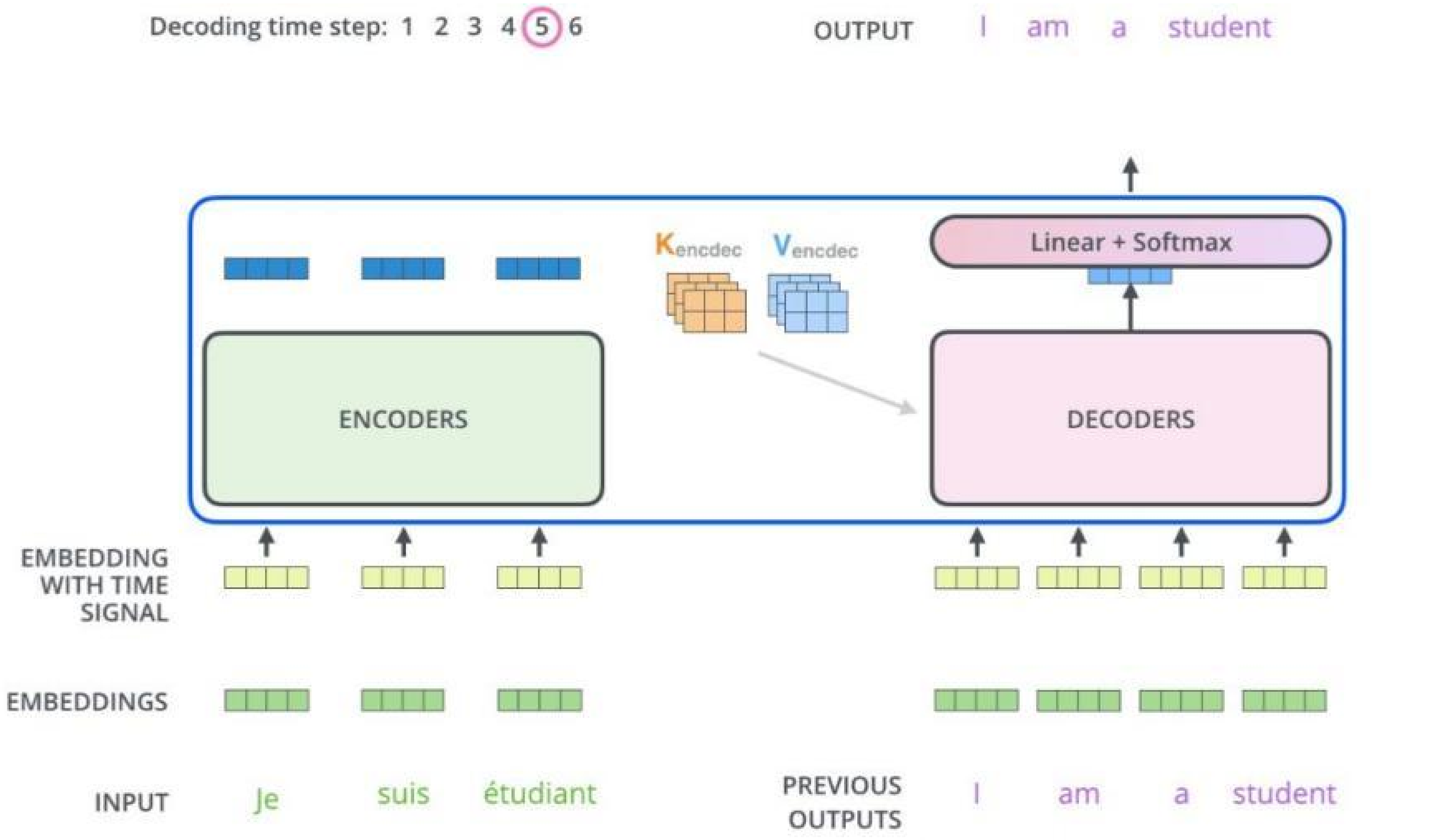
Transformer
Decoder



Transformer
Decoder

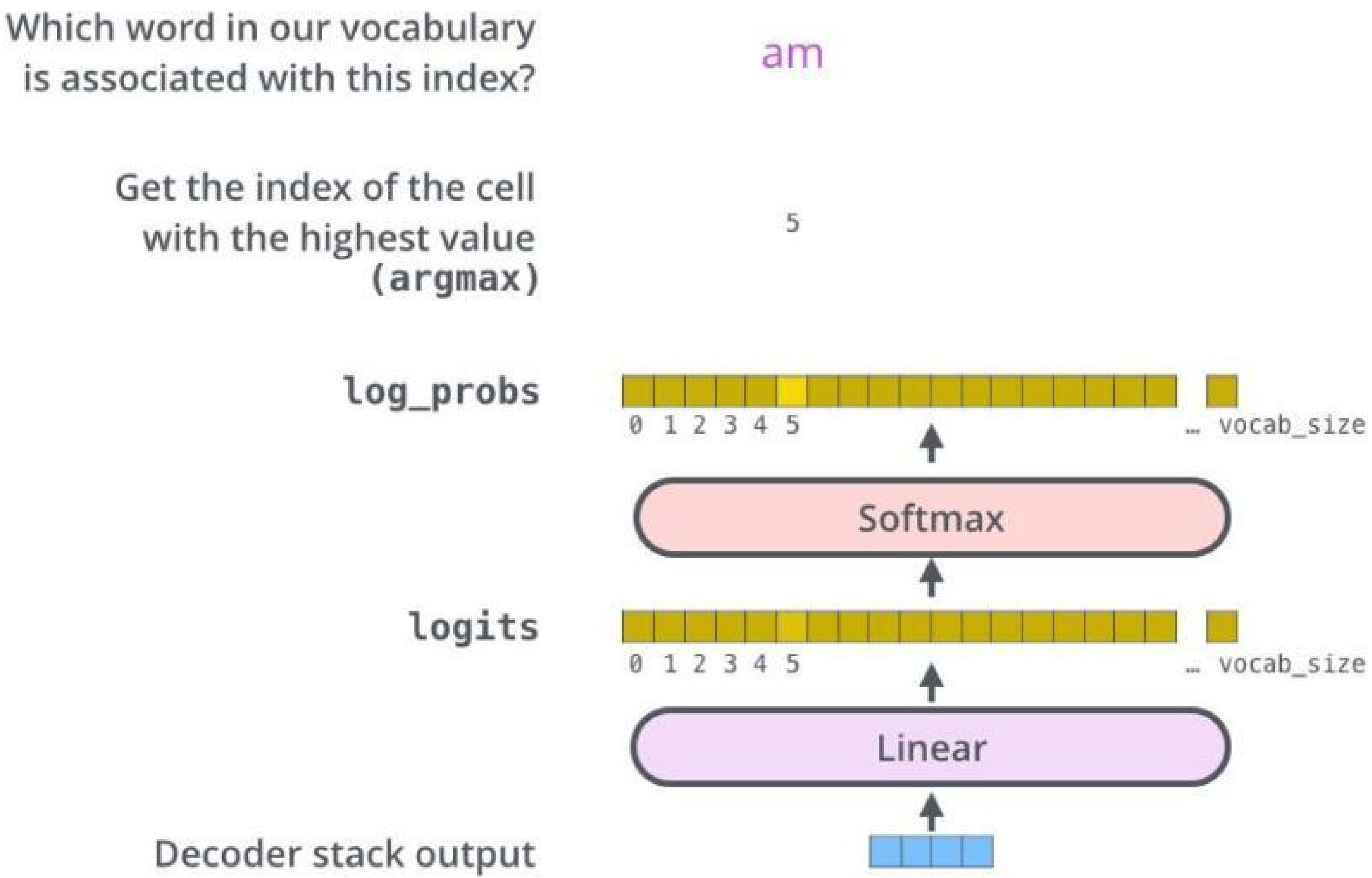


Transformer
Decoder



Transformer

Final Linear and Softmax Layer



Transformer
Training

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

Transformer
Training

Transformer

Training

Target Model Outputs

Output Vocabulary:	a	am	I	thanks	student	<eos>
position #1	0.0	0.0	1.0	0.0	0.0	0.0
position #2	0.0	1.0	0.0	0.0	0.0	0.0
position #3	1.0	0.0	0.0	0.0	0.0	0.0
position #4	0.0	0.0	0.0	0.0	1.0	0.0
position #5	0.0	0.0	0.0	0.0	0.0	1.0
	a	am	I	thanks	student	<eos>

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>

Transformer

多头注意力（Multi-headed attention）机制

1、 由编码器和解码器组成，在编码器的一个网络块中， 由一个多头attention子层和一个前馈神经网络子层组成，整个编码器栈式搭建了N个块。类似于编码器，只是解码器的一个网络块中多了一个多头attention层。为了更好的优化深度网络整个网络使用了残差连接和对层进行了规范化（Add&Norm）。

- Encoder and Decoder Stacks
  - **Attention**
  - Position-wise Feed-Forward Networks
  - **Positional Encoding**
  - Add & Norm

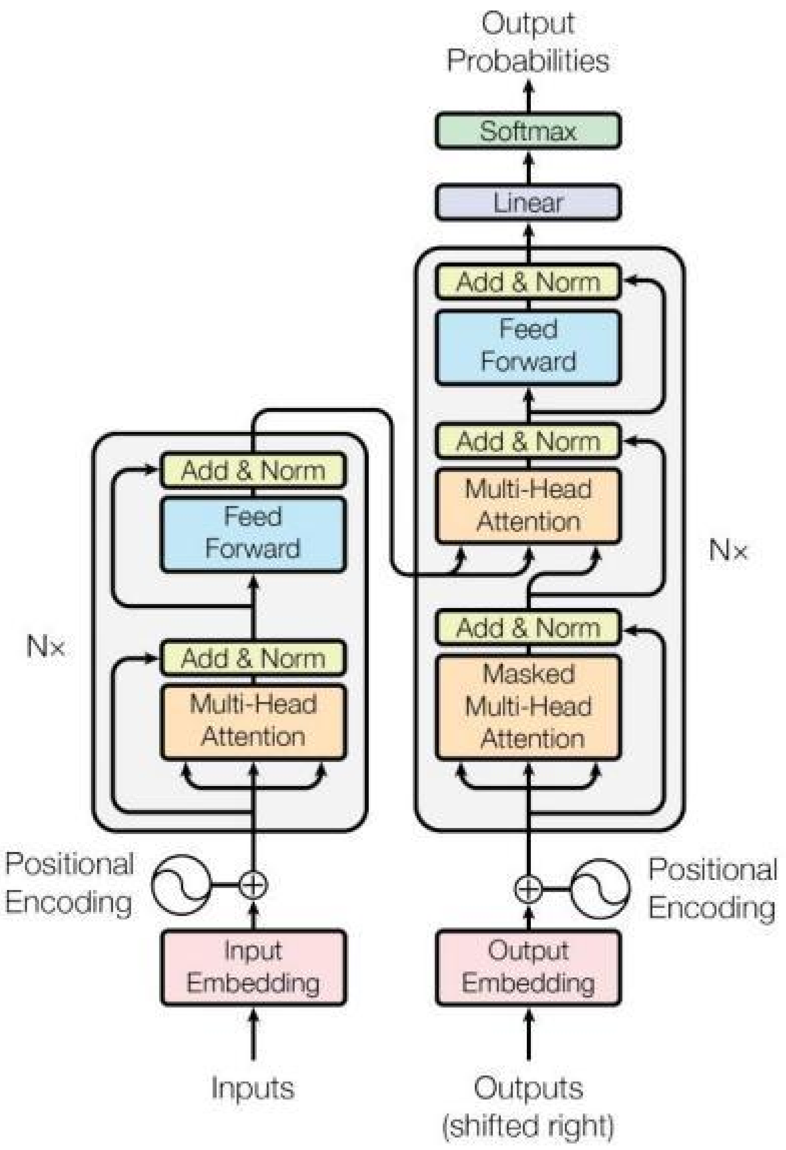


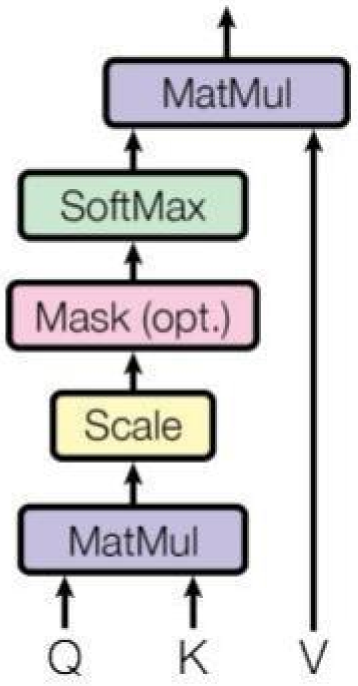
Figure 1: The Transformer - model architecture.

Transformer

2、放缩点积attention（scaled dot-Product attention）。对比我在前面背景知识里提到的attention的一般形式，其实scaled dot-Product attention就是我们常用的使用点积进行相似度计算的attention，只是多除了一个（为K的维度）起到调节作用，使得内积不至于太大。

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

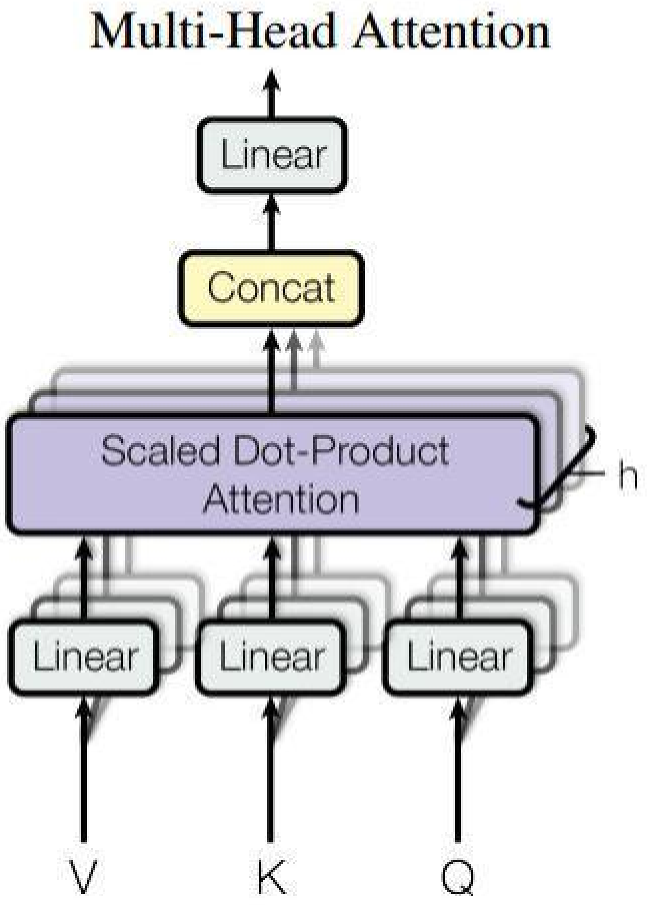
Scaled Dot-Product Attention



Transformer

3、 多头attention的Query， Key， Value首先进过一个线性变换， 然后输入到放缩点积attention， 注意这里要做h次， 其实也就是所谓的多头， 每一次算一个头 。而且每次Q， K， V进行线性变换的参数W是不一样的 。然后将h次的放缩点积attention结果进行拼接， 再进行一次线性变换得到的值作为多头attention的结果。

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



其它Transformer结构

- Weighted Transfomer：
  - https://arxiv.org/pdf/1711.02132.pdf
- Universal Transformer：
  - https://arxiv.org/pdf/1807.03819.pdf
- Gaussian Transformer
- IR Transformer
- NOTE:
  - https://blog.csdn.net/weixin_37947156/article/details/90112176

THANKS!