NLP项目
Transformer

课程内容

- Seq2Seq结构和Attention结构回顾
- Transformer结构讲 解

Seq2Seq结构回顾

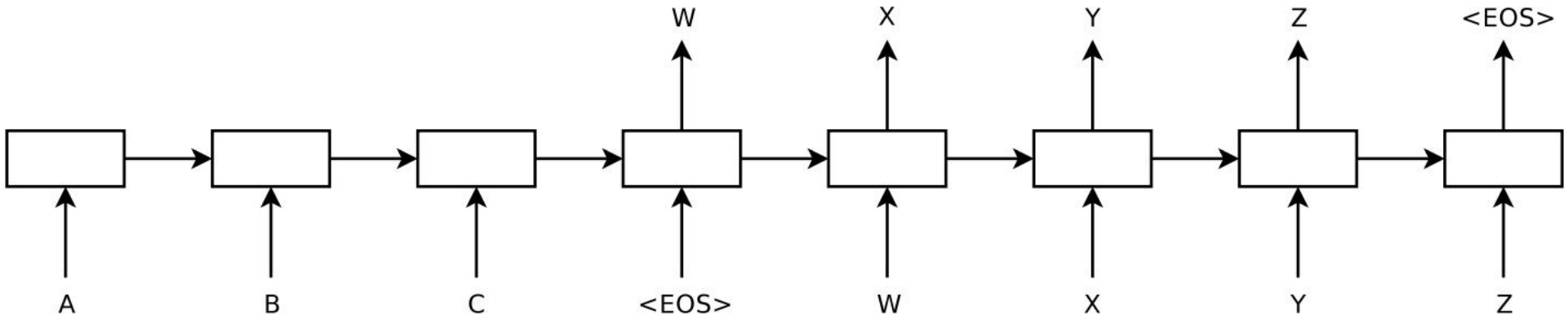
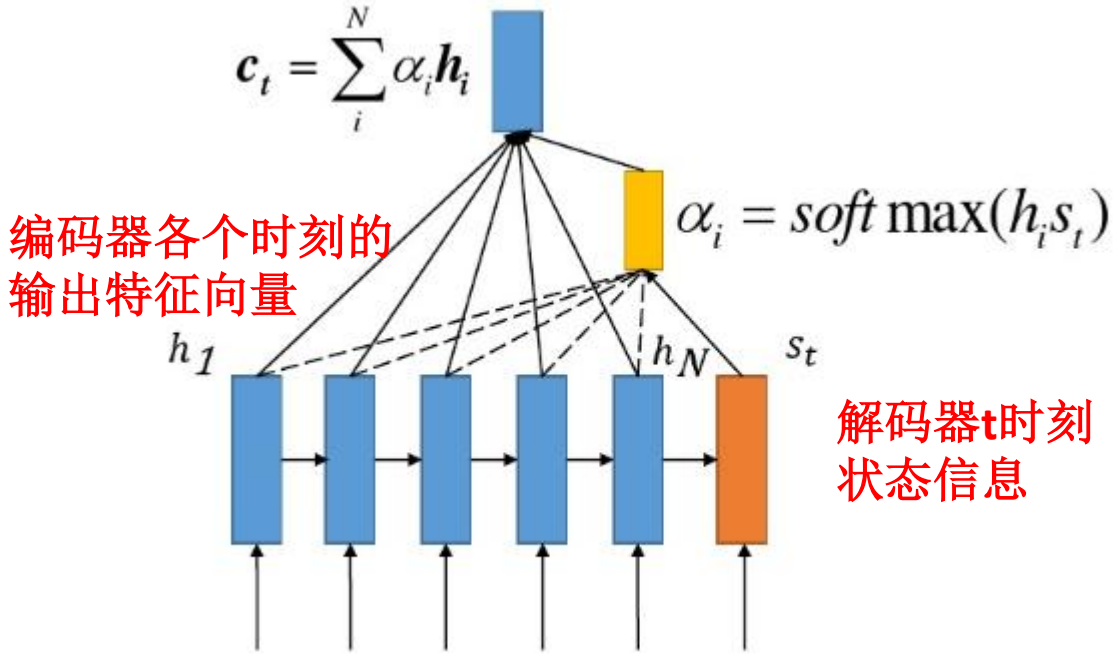


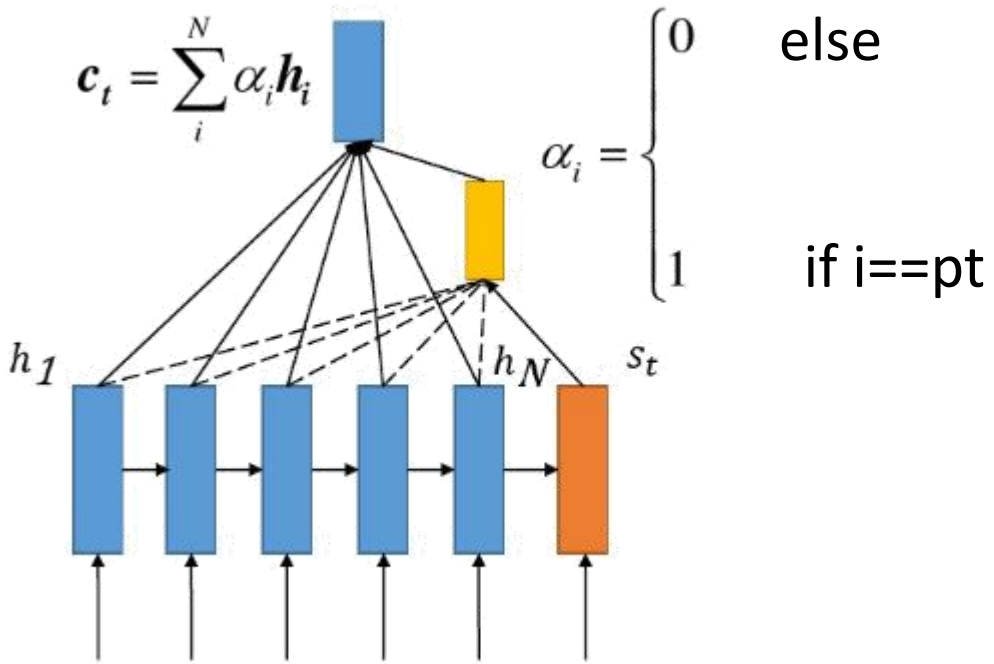
Figure 1: Our model reads an input sentence "ABC" and produces "WXYZ" as the output sentence. The model stops making predictions after outputting the end-of-sentence token. Note that the LSTM reads the input sentence in reverse, because doing so introduces many short term dependencies in the data that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

Attention结构回顾

等价到QKV结构中，此时的s_t就是Q，
h_1到h_n就是K和V(K和V是同一个向量)



Soft Attention



Hard Attention

Attention结构回顾

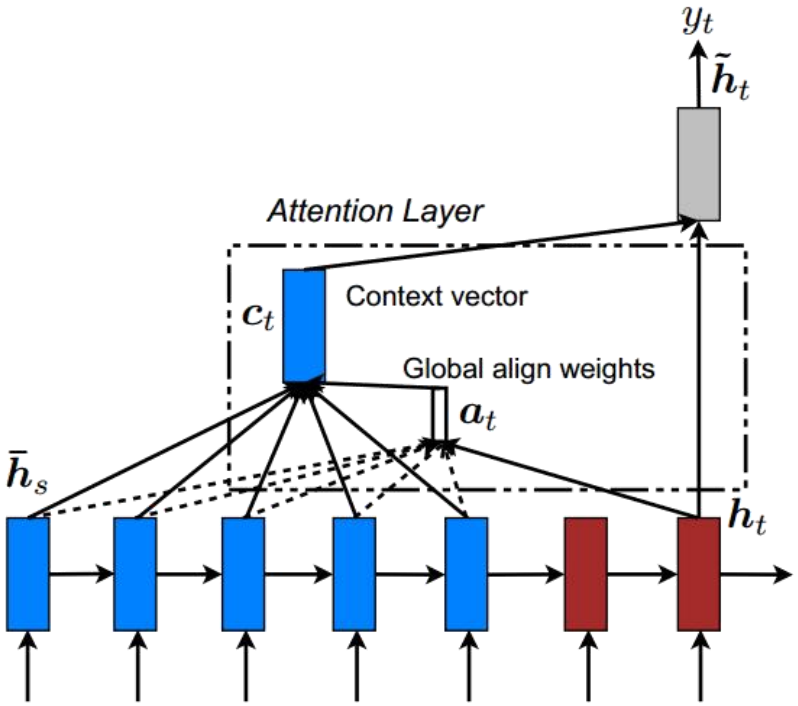


Figure 2: **Global attentional model** – at each time step t, the model infers a *variable-length* alignment weight vector $\boldsymbol{a}_t$ based on the current target state $\boldsymbol{h}_t$ and all source states $\bar{\boldsymbol{h}}_s$. A global context vector $\boldsymbol{c}_t$ is then computed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

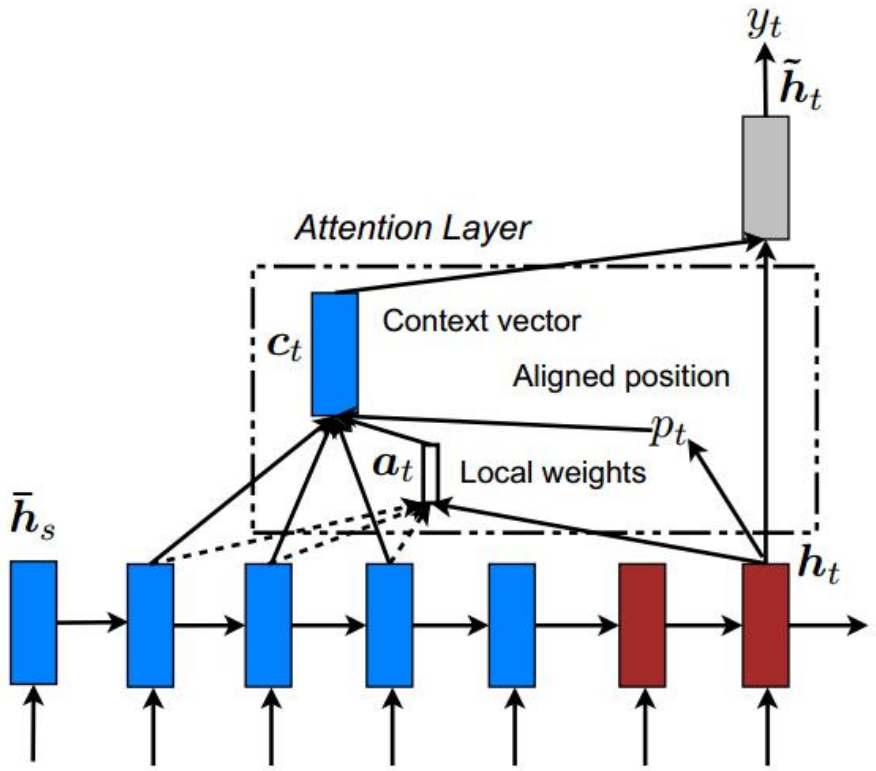
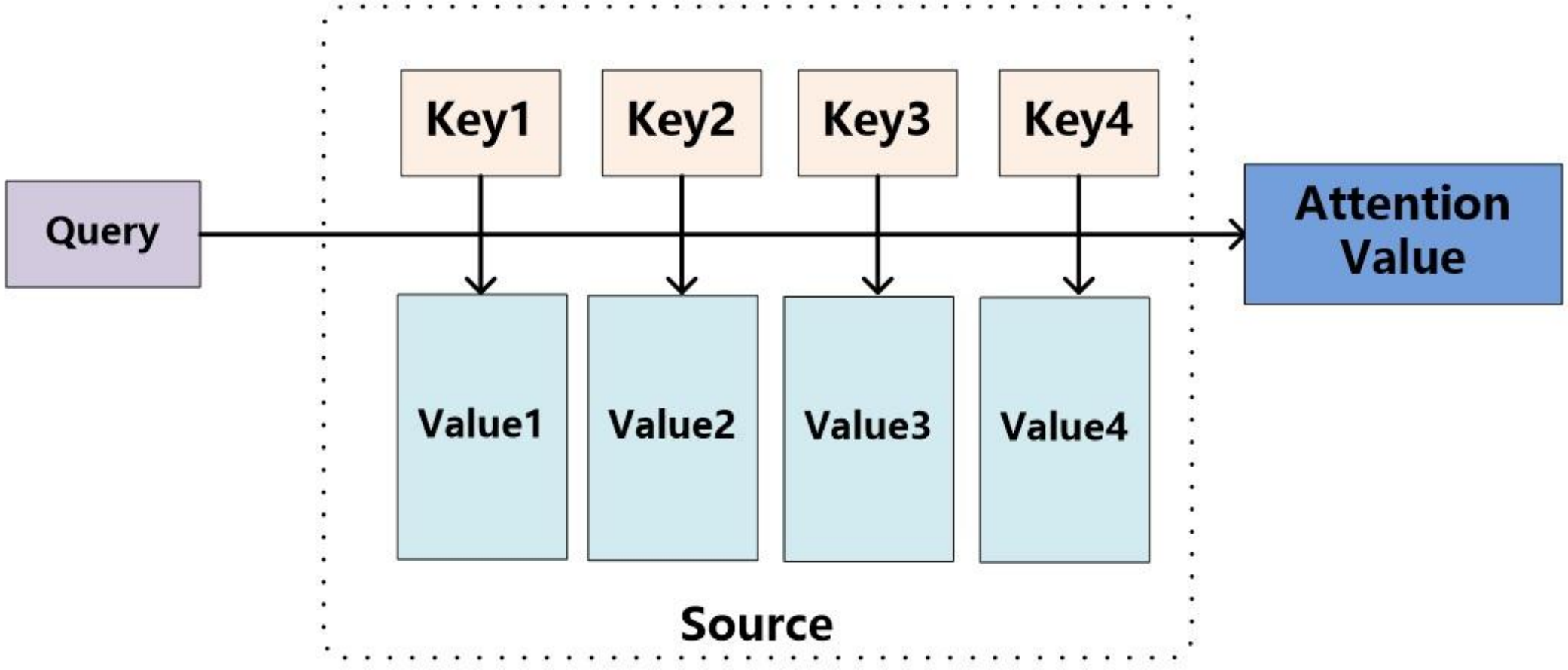


Figure 3: **Local attention model** – the model first predicts a single aligned position p_t for the current target word. A window centered around the source position p_t is then used to compute a context vector $\boldsymbol{c}_t$, a weighted average of the source hidden states in the window. The weights $\boldsymbol{a_t}$ 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的功能就相当于在将Value进行合并
---> 加权合并 ---> 有”偏向”的加权合并



Attention结构回顾

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

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

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

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

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

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

选择 $\frac{1}{\sqrt{d_k}}$ 作为Attention中缩放因子的原因，可以从数学上通过**高维空间中的内积的分布**来理解。我们可以从以下几个角度进行分析：

1. 点积的预期值和方差

假设我们有两个独立的随机向量 $\mathbf{q}$ 和 $\mathbf{k}$，它们的维度为 d_k，并且每个维度的分量都是独立的零均值 $(\mathbb{E}[q_i] = \mathbb{E}[k_i] = 0)$ 且方差为1 $(\mathrm{Var}(q_i) = \mathrm{Var}(k_i) = 1)$ 。我们来看它们的点积 $\mathbf{q} \cdot \mathbf{k}$：

$$\mathbf{q} \cdot \mathbf{k} = \sum_{i=1}^{d_k} q_i \cdot k_i$$

由于每个 q_i 和 k_i 都是独立的零均值随机变量，且它们的方差为1，点积的期望值是：

$$\mathbb{E}[\mathbf{q} \cdot \mathbf{k}] = \mathbb{E}\left[\sum_{i=1}^{d_k} q_i \cdot k_i\right] = \sum_{i=1}^{d_k} \mathbb{E}[q_i \cdot k_i] = 0$$

点积的方差为：

$$\mathrm{Var}(\mathbf{q} \cdot \mathbf{k}) = \mathbb{E}[(\mathbf{q} \cdot \mathbf{k})^2] - (\mathbb{E}[\mathbf{q} \cdot \mathbf{k}])^2$$

由于 $\mathbb{E}[\mathbf{q} \cdot \mathbf{k}] = 0$，我们只需要计算 $\mathbb{E}[(\mathbf{q} \cdot \mathbf{k})^2]$：

$$\mathbb{E}[(\mathbf{q} \cdot \mathbf{k})^2] = \mathbb{E}\left[\left(\sum_{i=1}^{d_k} q_i \cdot k_i\right)^2\right] = \mathbb{E}\left[\sum_{i=1}^{d_k} q_i^2 \cdot k_i^2 + 2\sum_{i<j} q_i \cdot k_i \cdot q_j \cdot k_j\right]$$

由于 q_i 和 k_i 是独立的，且方差为1，得到：

$$\mathbb{E}[q_i^2] = \mathbb{E}[k_i^2] = 1$$

而跨项 $(i \neq j)$ 的期望为0，因此：

$$\mathbb{E}[(\mathbf{q} \cdot \mathbf{k})^2] = \sum_{i=1}^{d_k} \mathbb{E}[q_i^2]\mathbb{E}[k_i^2] = d_k$$

2. 为什么要缩放点积?

当我们计算多个高维向量的点积时，随着维度 d_k 的增大，点积的方差会变得越来越大。这样，如果直接使用点积的结果，它会因为维度的增大而变得越来越大，这使得模型的梯度更新变得不稳定，尤其是在训练时。

为了保持点积的规模稳定，我们通过除以 $\sqrt{d_k}$ 来缩放点积的结果。这是因为：

$$\mathrm{Var}\left(\frac{1}{\sqrt{d_k}}\mathbf{q} \cdot \mathbf{k}\right) = \frac{1}{d_k} \cdot \mathrm{Var}(\mathbf{q} \cdot \mathbf{k}) = \frac{1}{d_k} \cdot d_k = 1$$

这样，缩放后的点积的方差就变成了1，这样可以避免点积随着维度增大而变得过大，确保训练过程中的数值稳定性。

3. 直观理解

- **维度增大导致方差增大**：随着维度 d_k 增大，两个随机向量的点积变得更加"集中"在0附近，但方差会增大。因此，不对点积进行缩放的话，得到的注意力权重（经过softmax的结果）会非常集中，导致梯度更新时可能出现过大的变化。
- **除以 $\sqrt{d_k}$**：这个缩放因子使得点积在高维空间中也能保持稳定的方差和尺度，从而避免了因为维度增大导致的不稳定数值问题。

4. 结论

从数学上讲，选择 $\frac{1}{\sqrt{d_k}}$ 是为了将点积的方差控制在一个稳定的范围内，避免随着向量维度的增大，点积的方差无限增大。这是为了保证模型的稳定性，特别是在训练过程中，避免梯度爆炸或数值不稳定的现象。因此，根号 d_k 是对高维空间中内积的自然缩放，确保它在训练过程中不会过于剧烈变化。

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

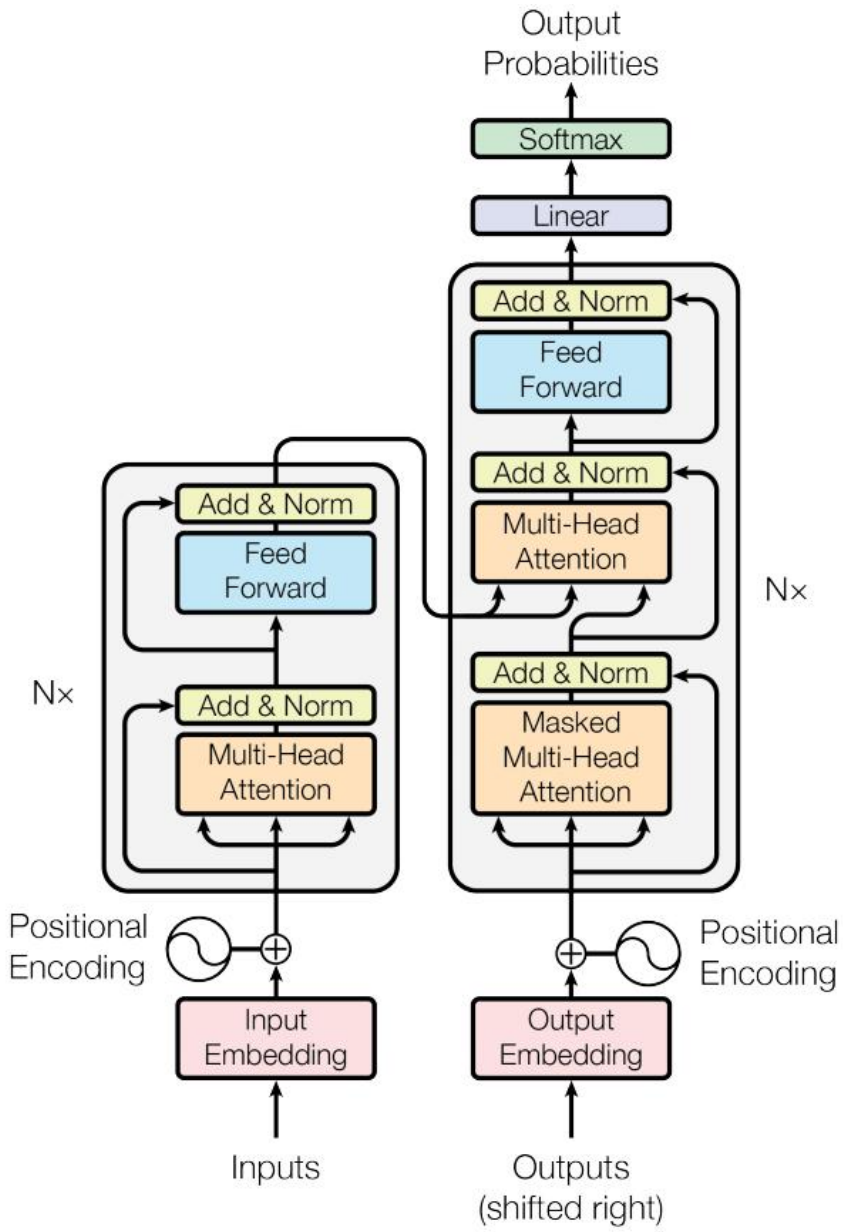


Figure 1: The Transformer - model architecture.

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：** **为了提取token的特征向量；**
    - **Multi-Headed-Attention：** **为了提取不同”角度“的特征向量；**
    - **Positional Encoding：** **位置编码，为了解决attention结构没有序列特性提取的能力；**
    - **Residuals：** **残差结构，为了防止模型退化；**
    - **Layer Norm：** **归一化，为了防止模型过拟合；**
    - **Masked：** **提取token特征向量的时候，考虑方向(正向)；**
    - **Feed Forward（FFN）：** **全连接，进一步提取高阶特征。**

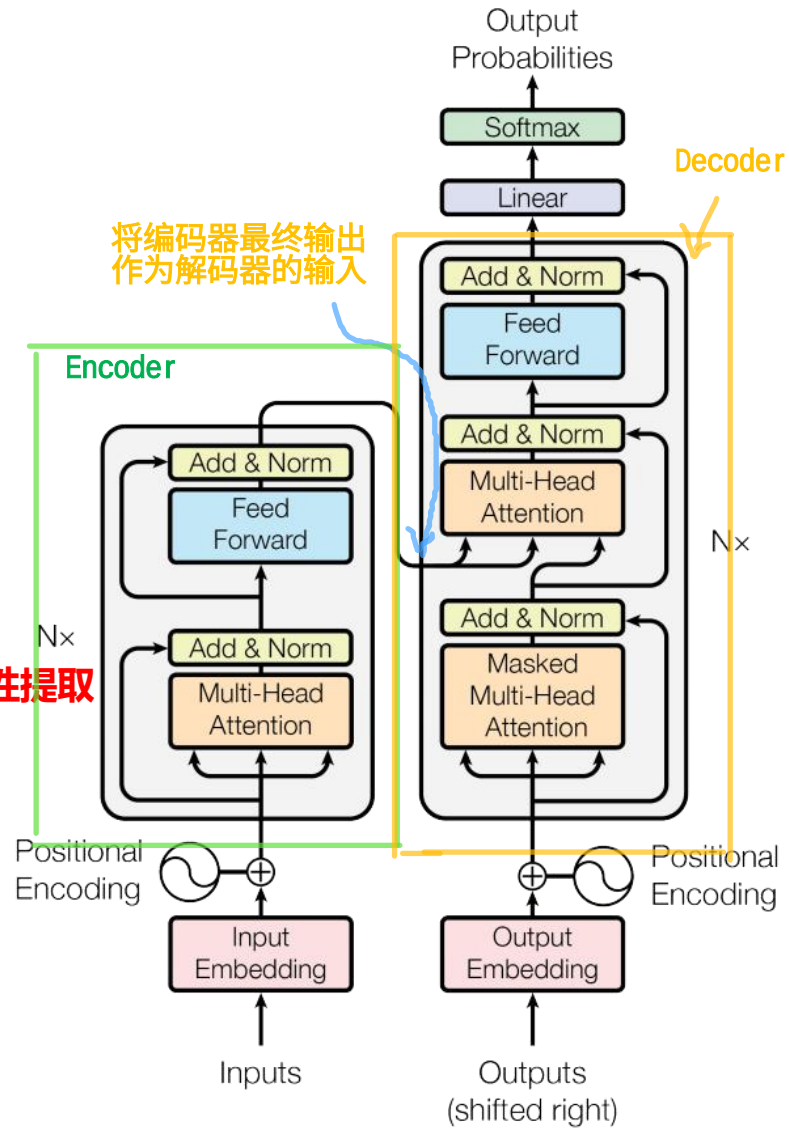
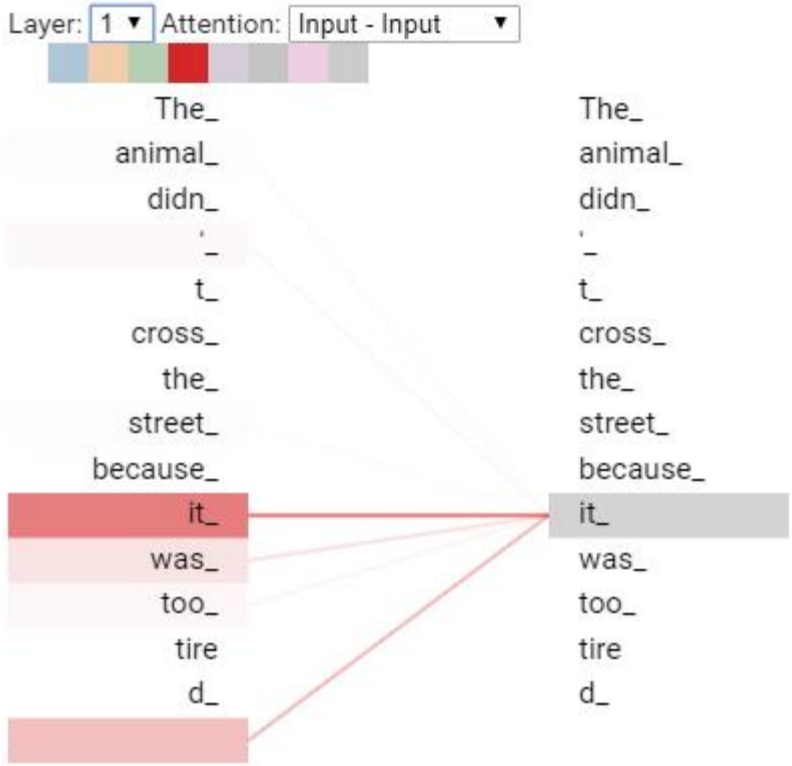
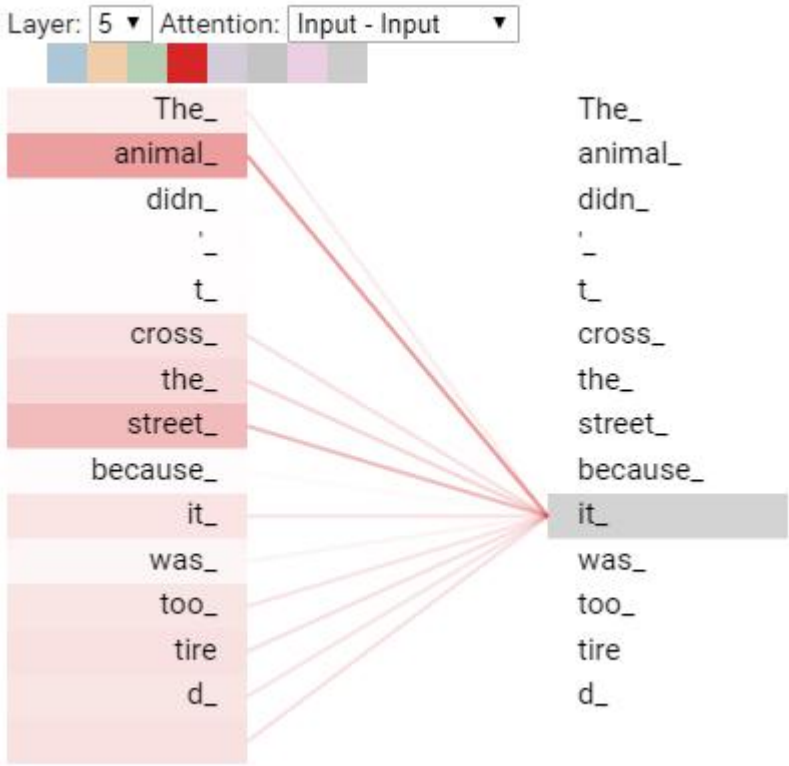


Figure 1: The Transformer - model architecture.

Transformer

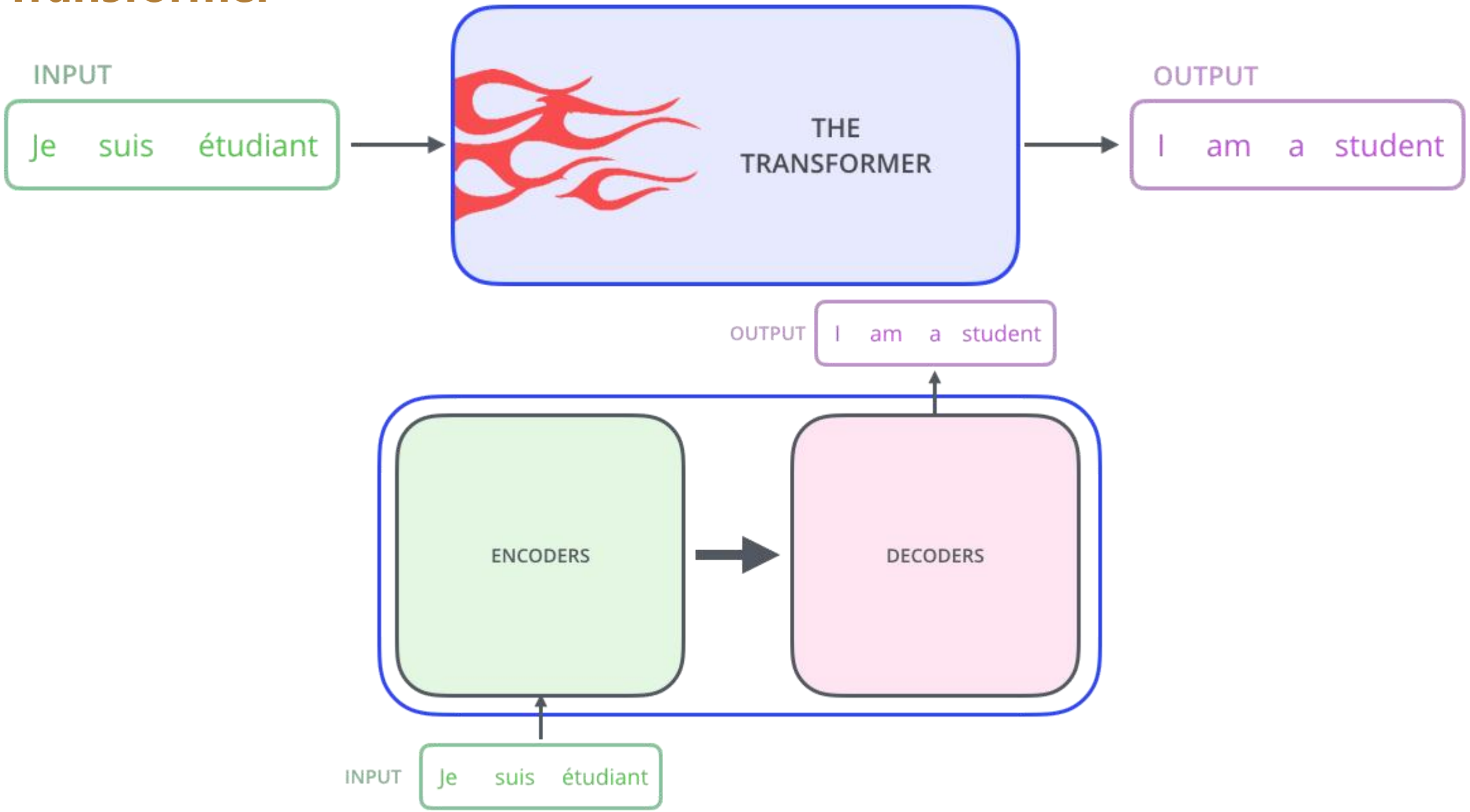
https://colab.research.google.com/github/tensorflow/tensor2tensor/blob/master/tensor2tensor/notebooks/hello_t2t.ipynb#scrollTo=OJKU36QAfqOC





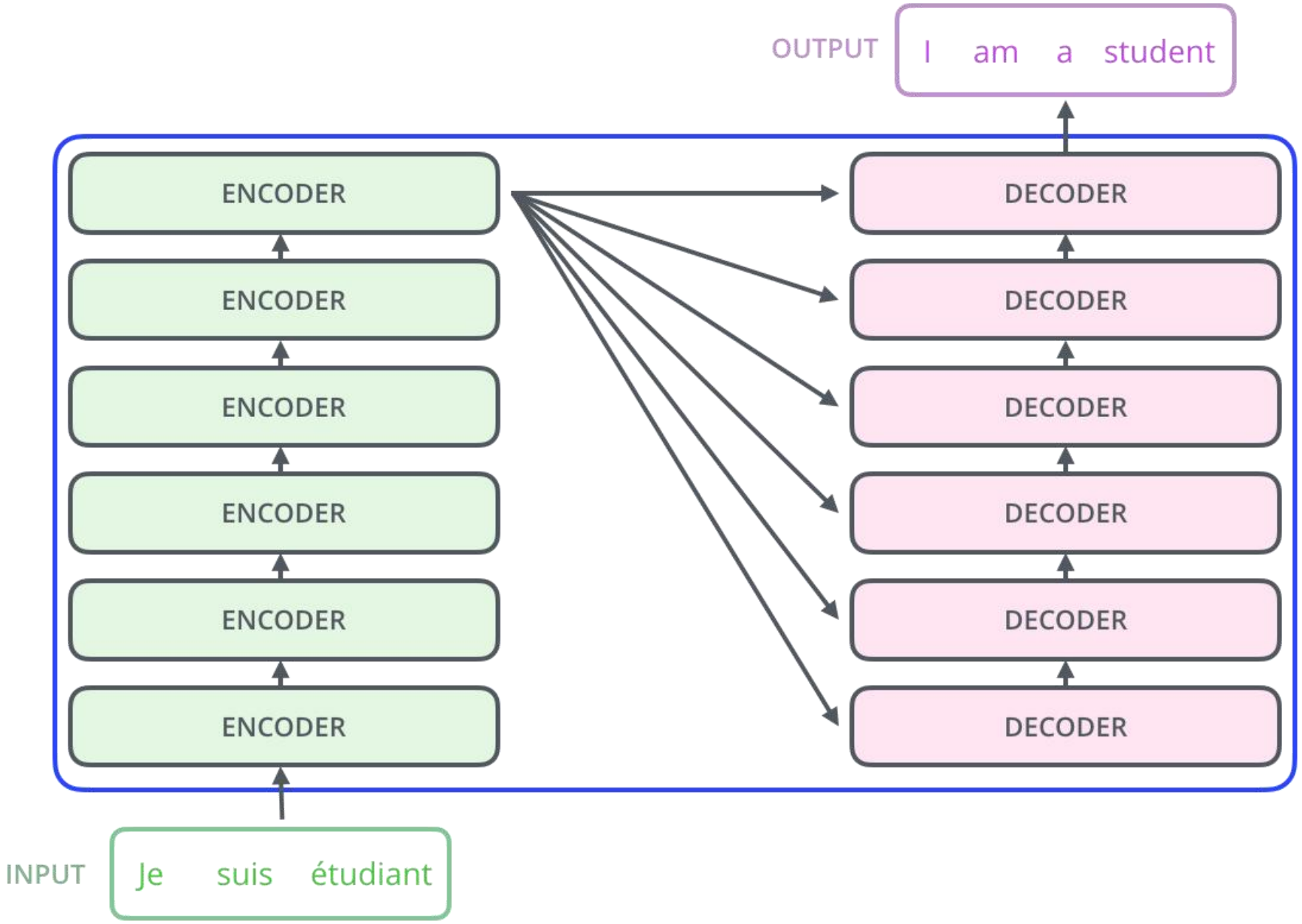
The animal didn't cross the street because it was too tired

Transformer



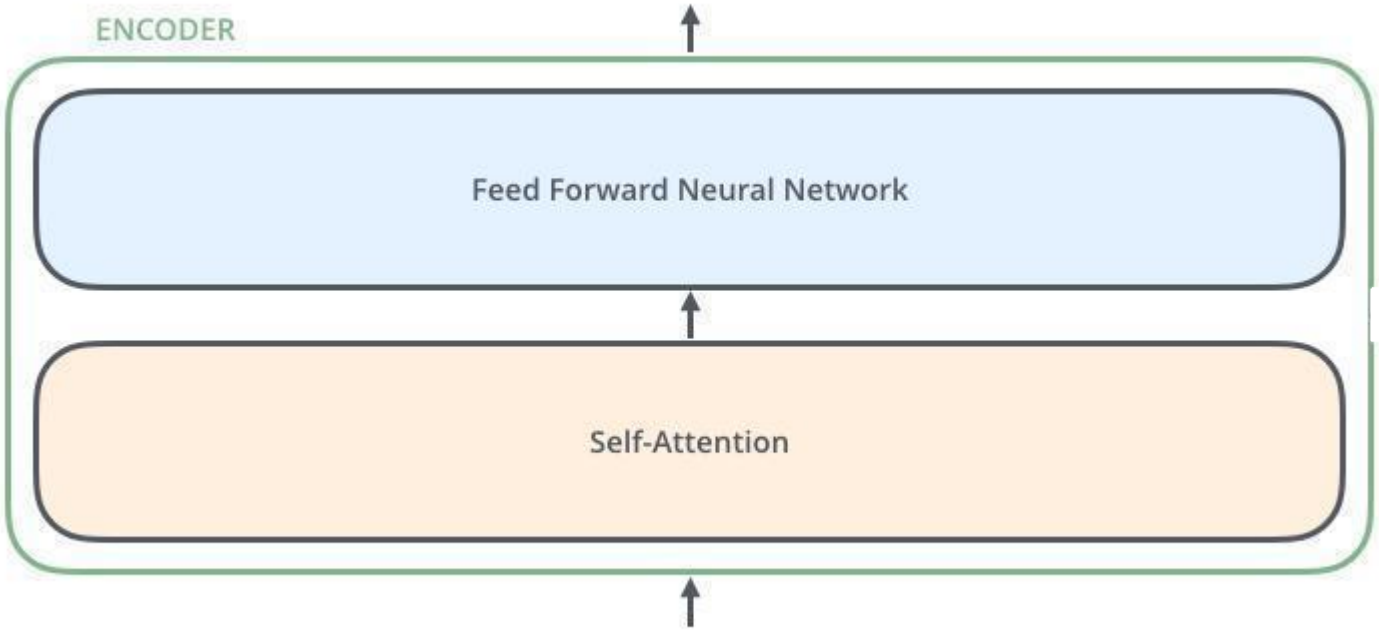
Transformer

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



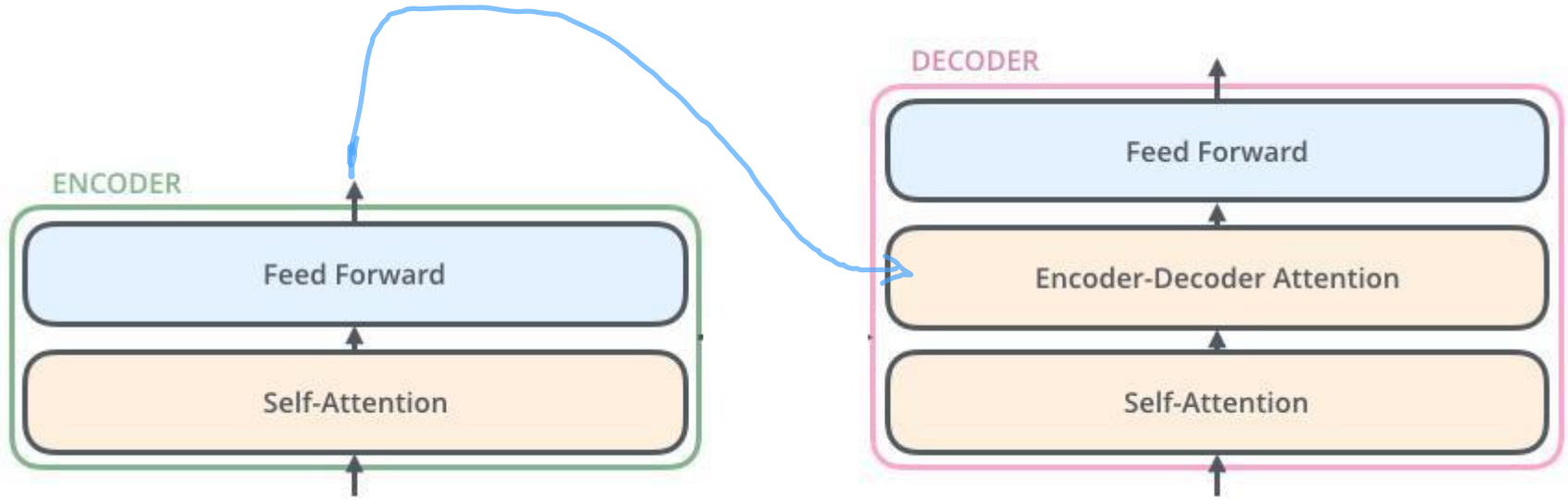
Transformer

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

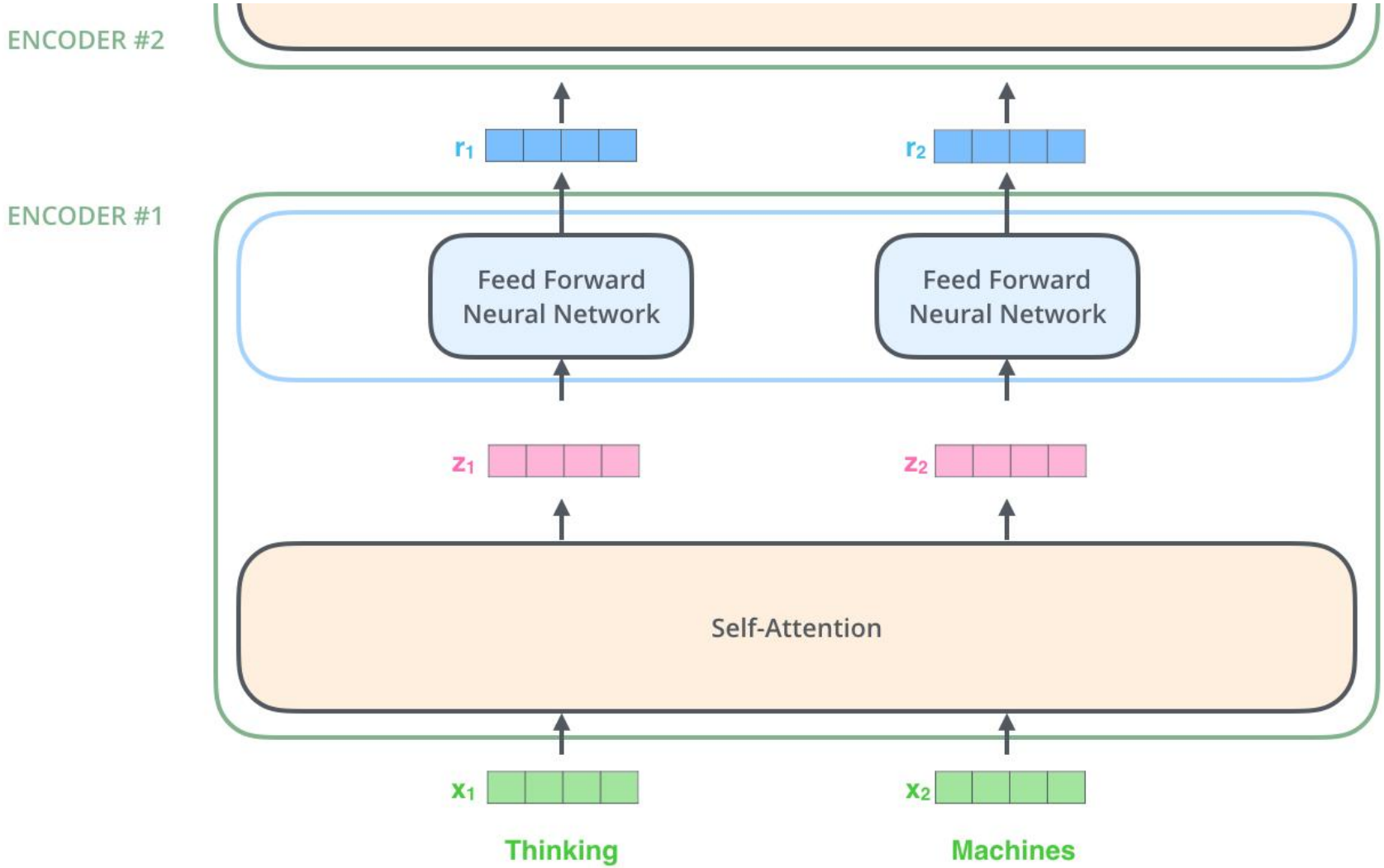


Transformer

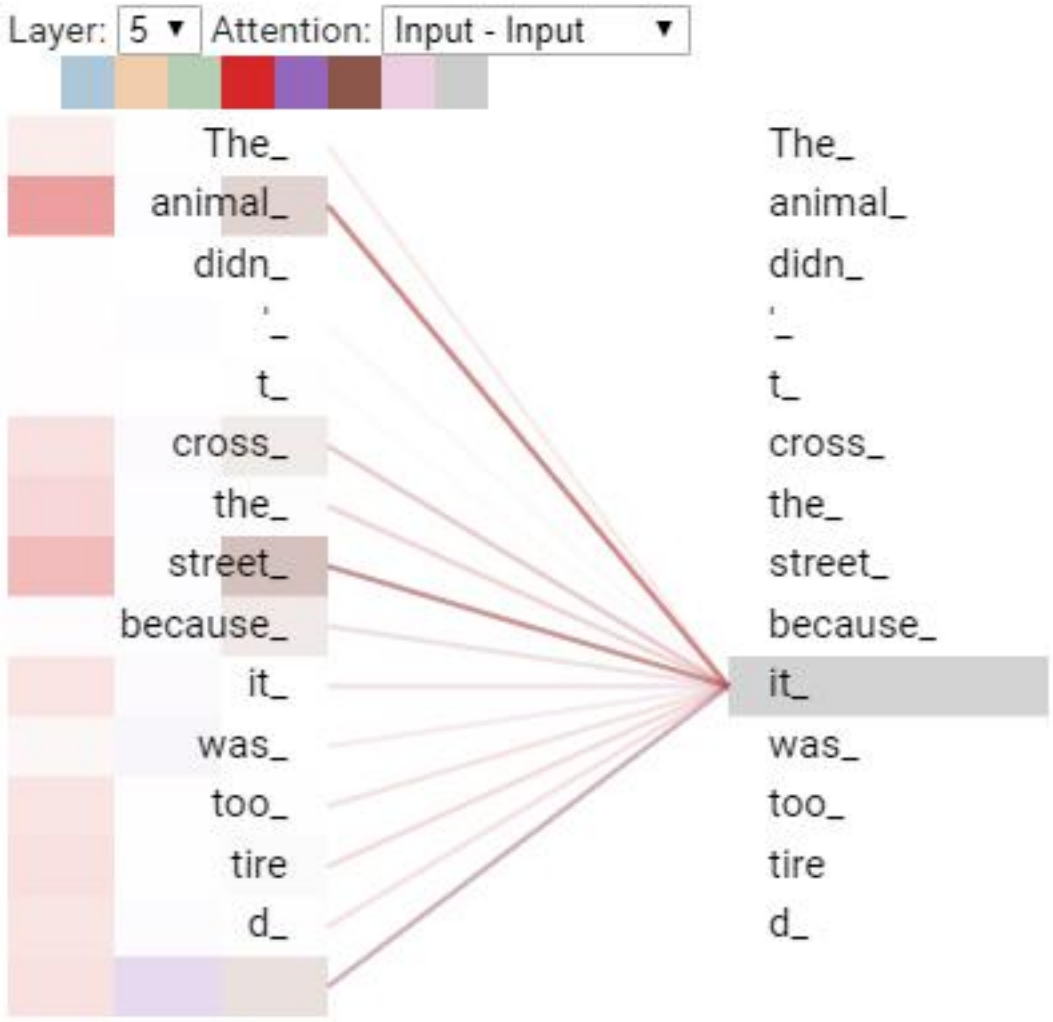
每个Decoder在Encoder的基础上，在Self-Attention和feed-forward NN之间增加了一个Encoder-Decoder Attention



Transformer
Encoder

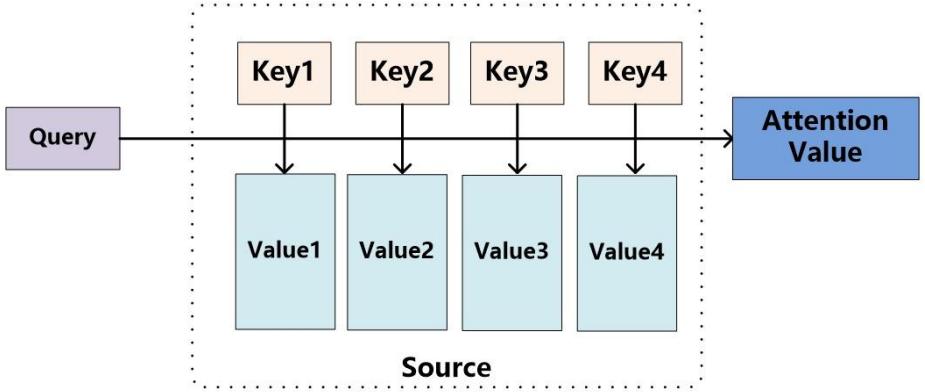


Transformer
Encoder Self-Attention



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

Transformer
Encoder Self-Attention



Input — Thinking — Machines

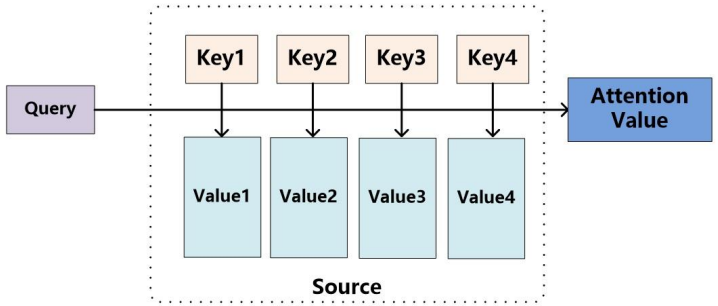
Embedding — X_1 — X_2

Queries — q_1 — q_2 — W^Q — $q = XW^Q$

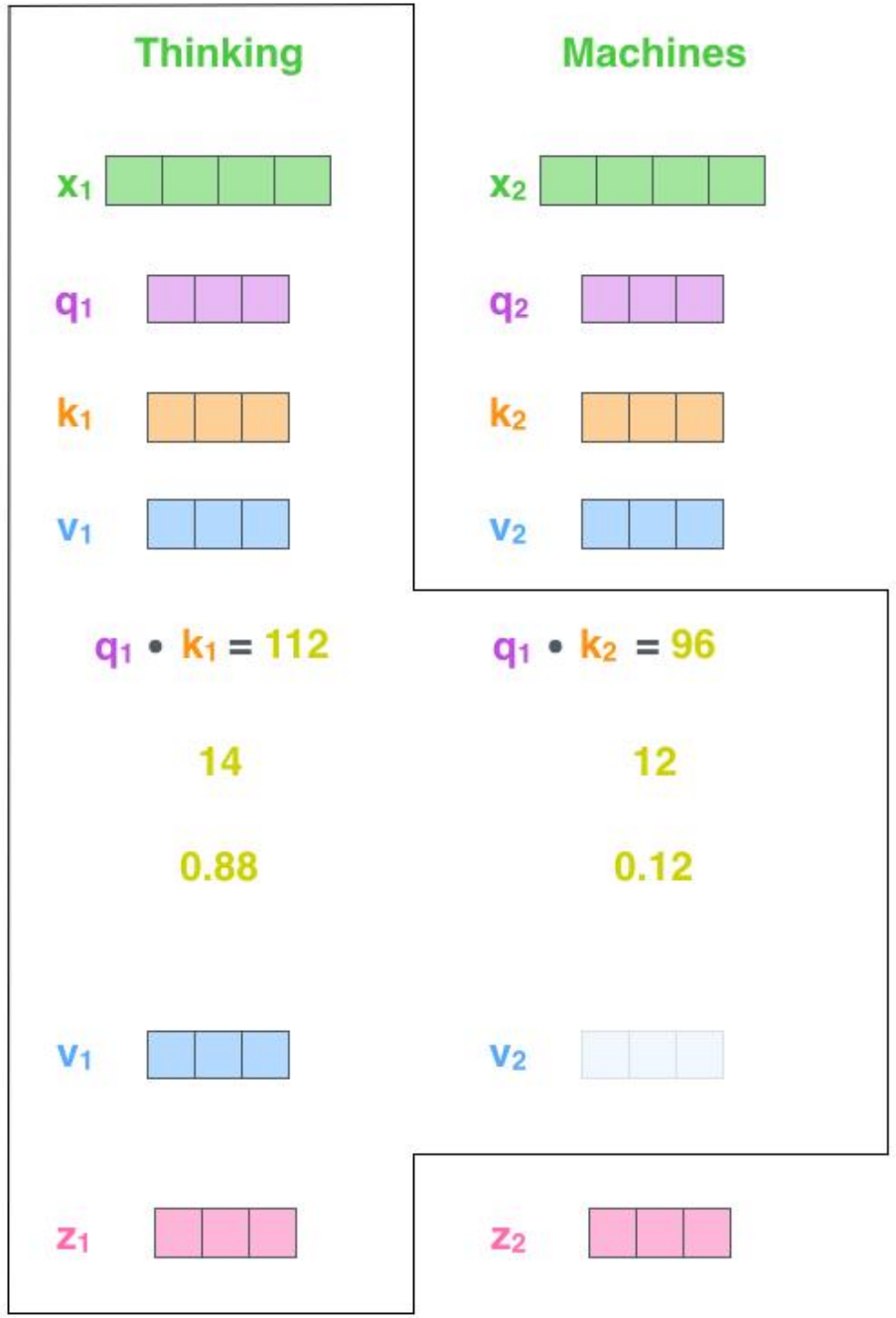
Keys — k_1 — k_2 — W^K — $k = XW^K$

Values — v_1 — v_2 — W^V — $v = XW^V$

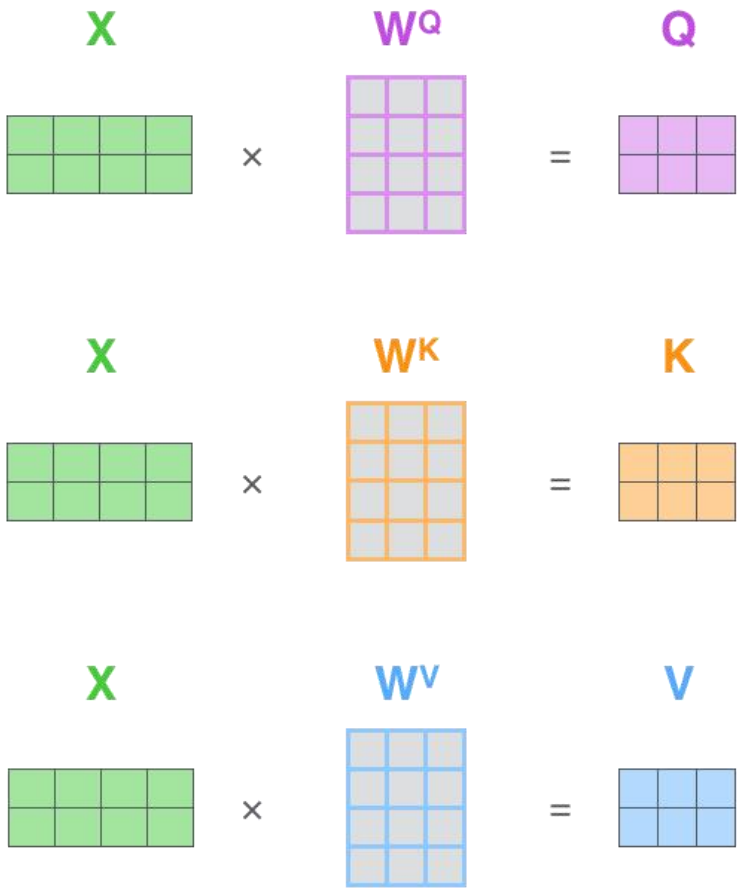
Transformer
Encoder Self-Attention



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

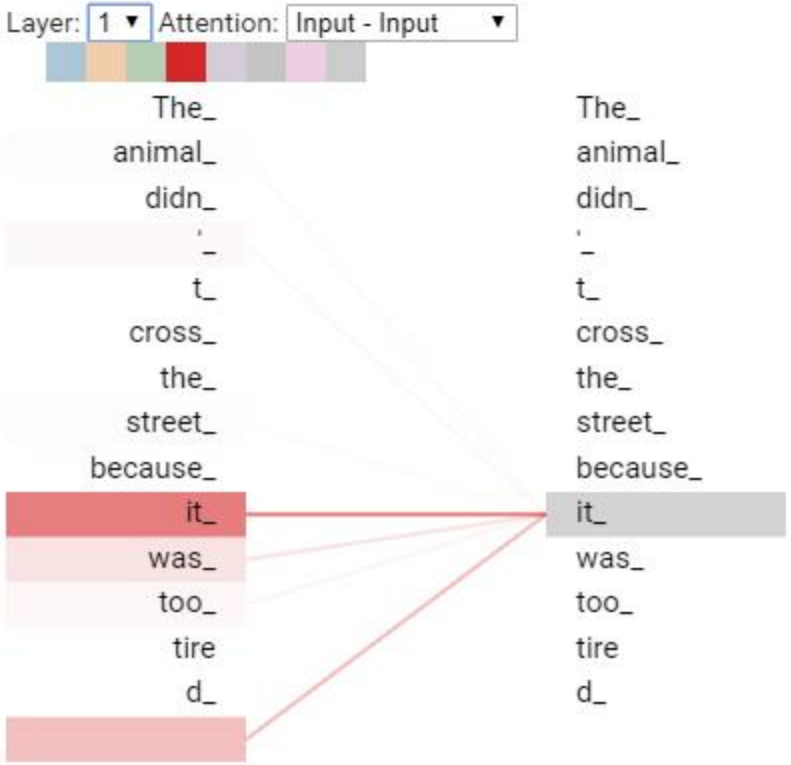


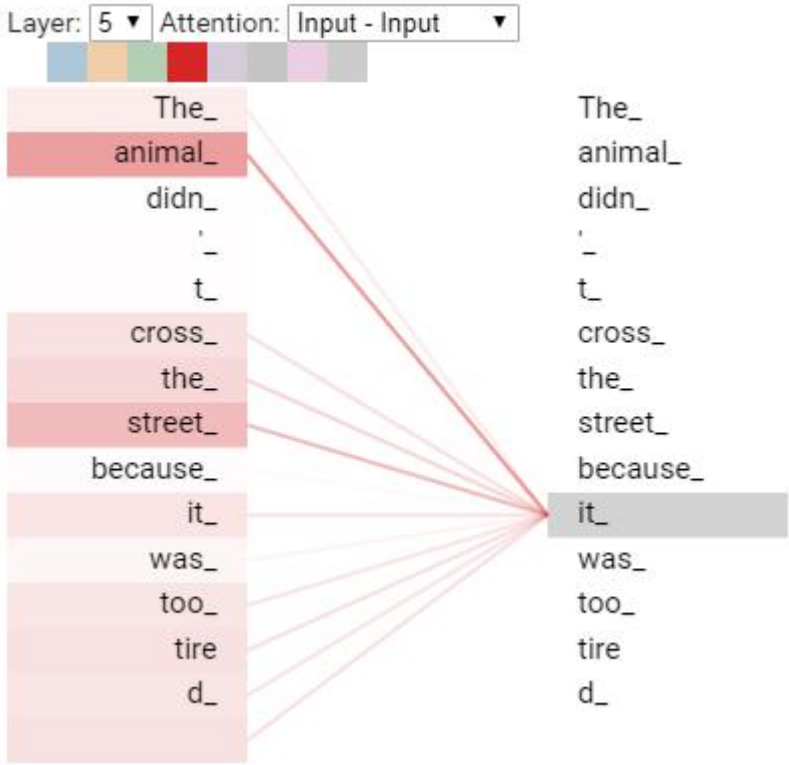
Transformer
Encoder Self-Attention



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

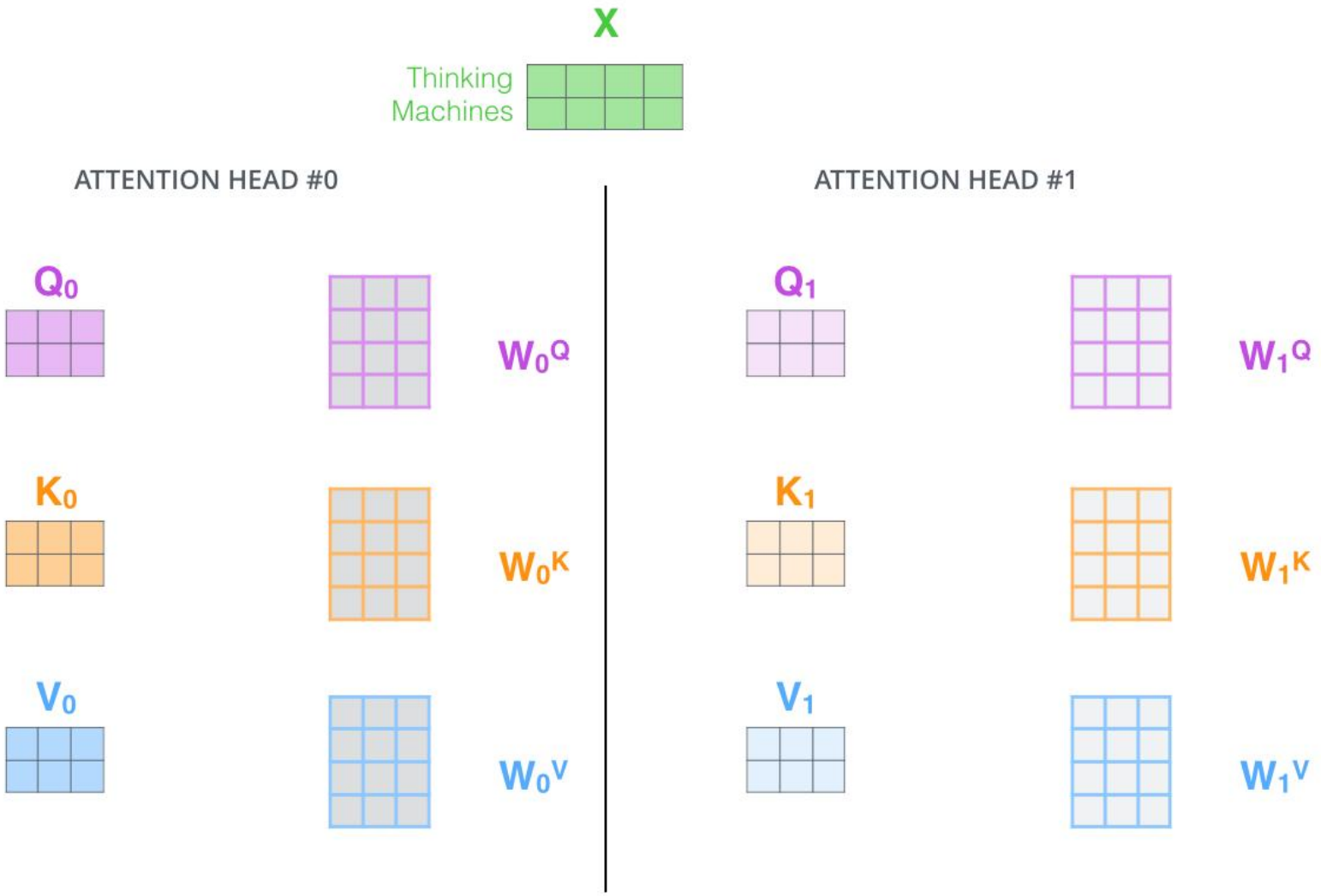
Transformer
Encoder Self-Attention



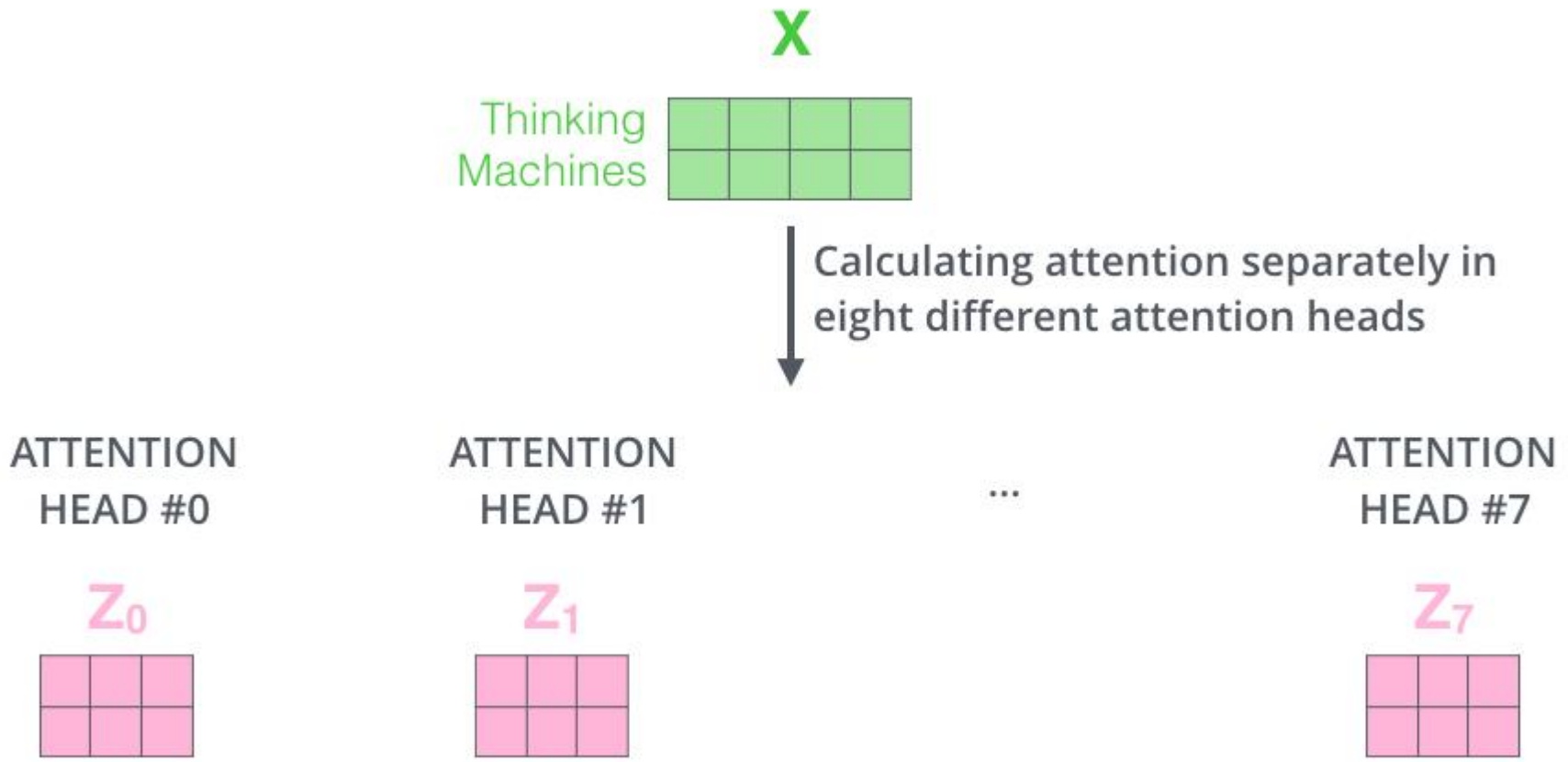


Transformer
Encoder Multi-Headed-Attention

多头Attention：实际上就是采用不同的参数来提取不同的特征



Transformer
Encoder Multi-Headed-Attention

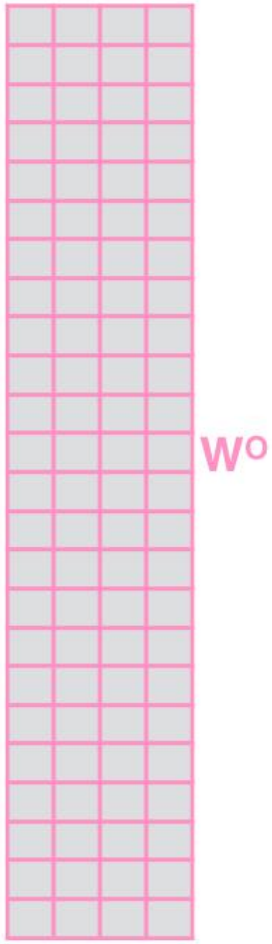


Transformer
Encoder Multi-Headed-Attention

1) Concatenate all the attention heads

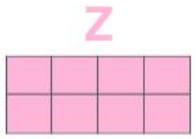
2) Multiply with a weight matrix W^O that was trained jointly with the model

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

1) This is our input sentence*

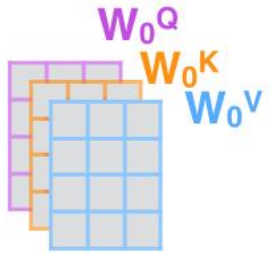
2) We embed each word*

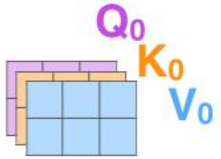
3) Split into 8 heads. We multiply X or R with weight matrices

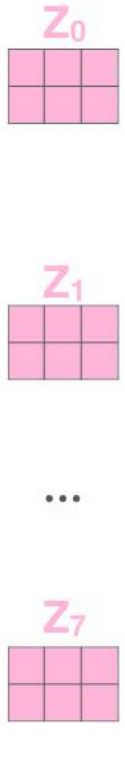
4) Calculate attention using the resulting Q/K/V matrices

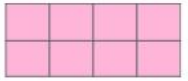
5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer

Thinking
Machines



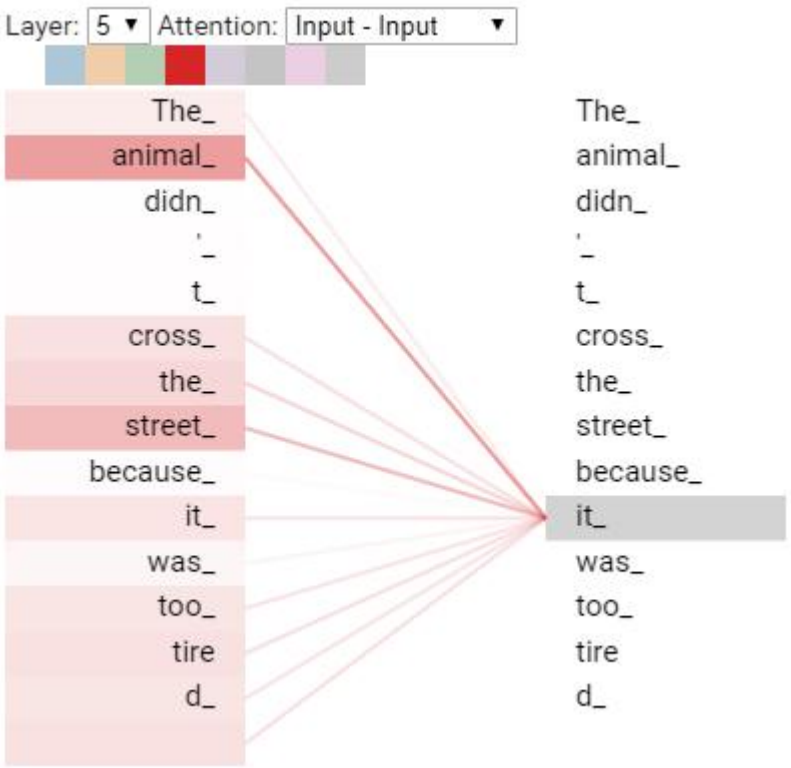


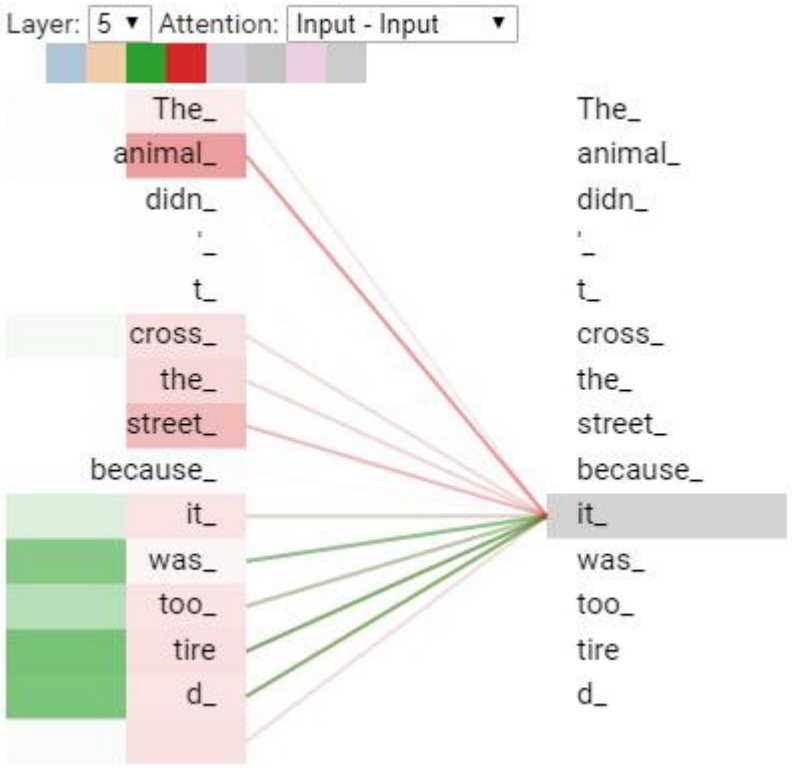




* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one

Transformer
Encoder Multi-Headed-Attention





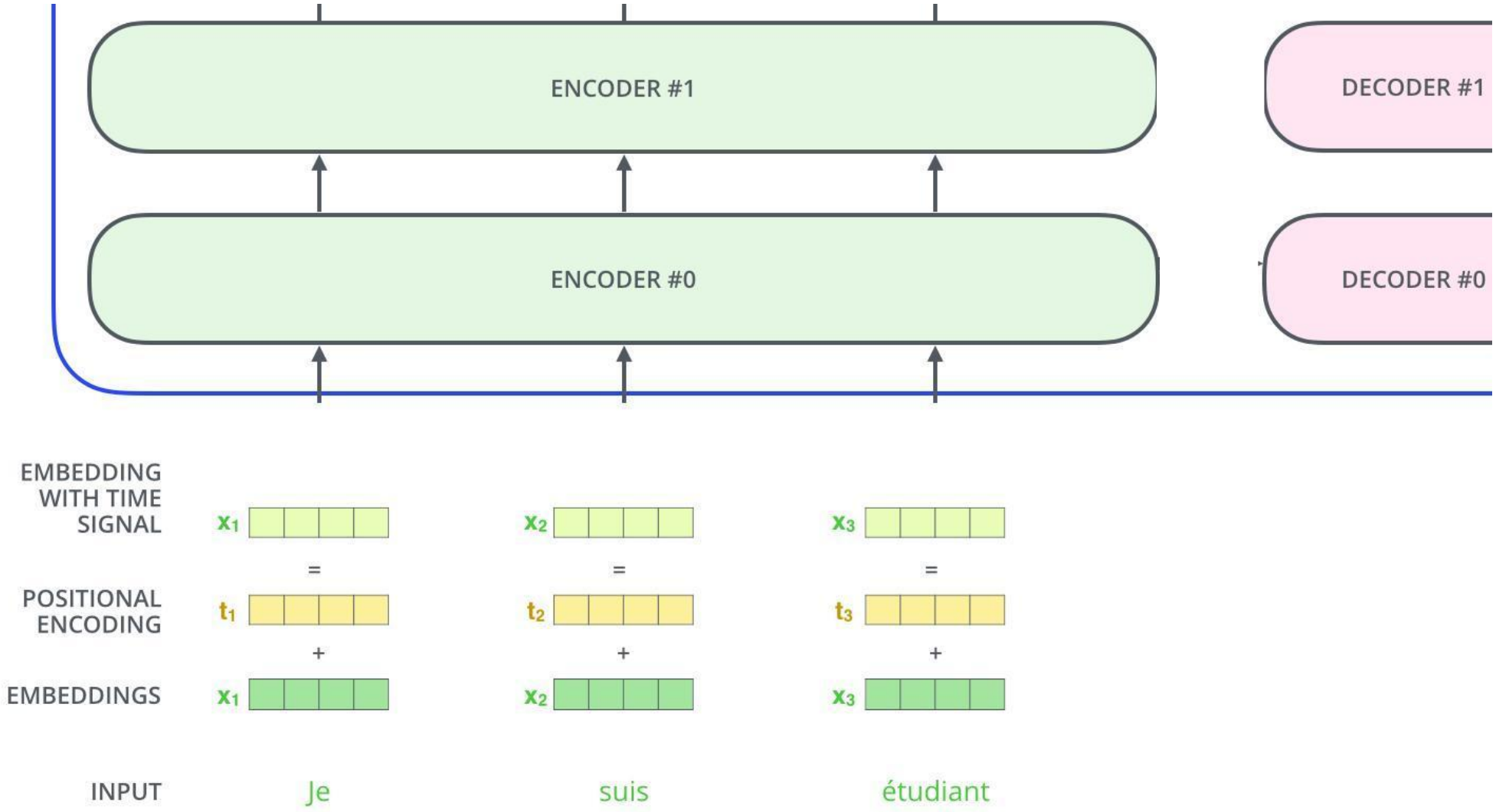
Transformer
Positional Encoding

- 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(**绝对位置编码**)，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。

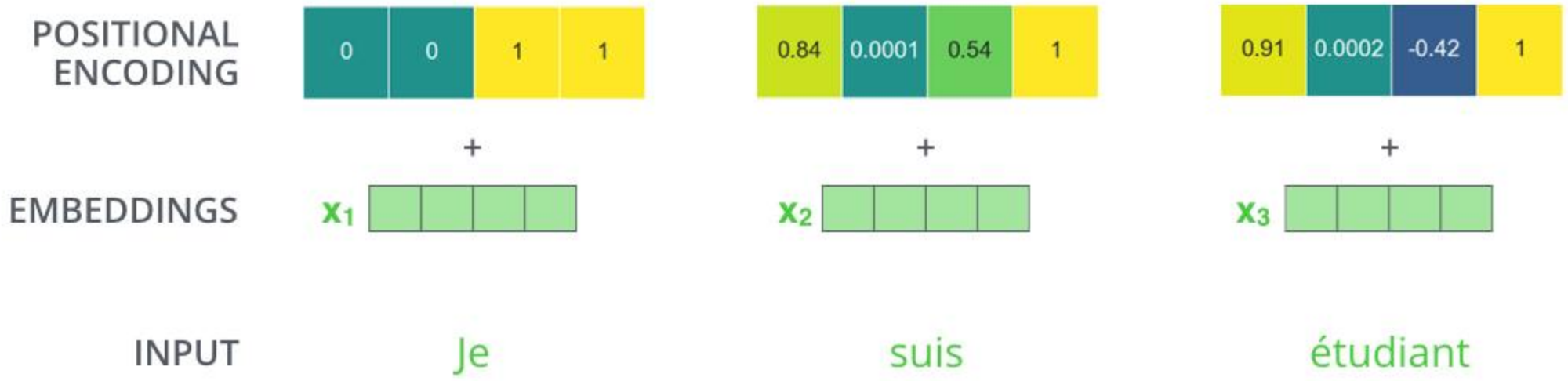
$$PE(pos, 2i) = sin(\frac{pos}{10000^{\frac{2i}{d_{model}}}})$$

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

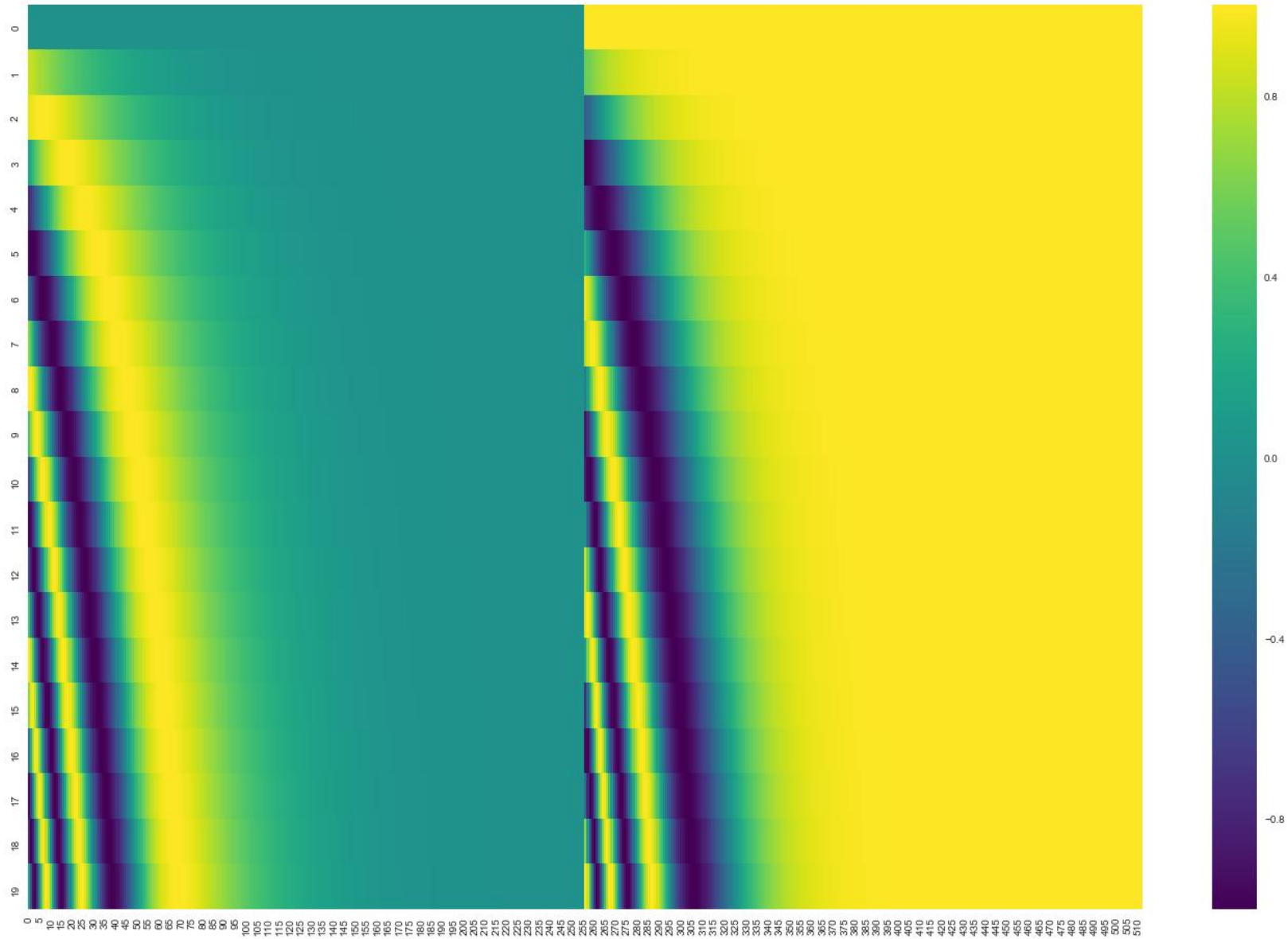
Transformer
Positional Encoding



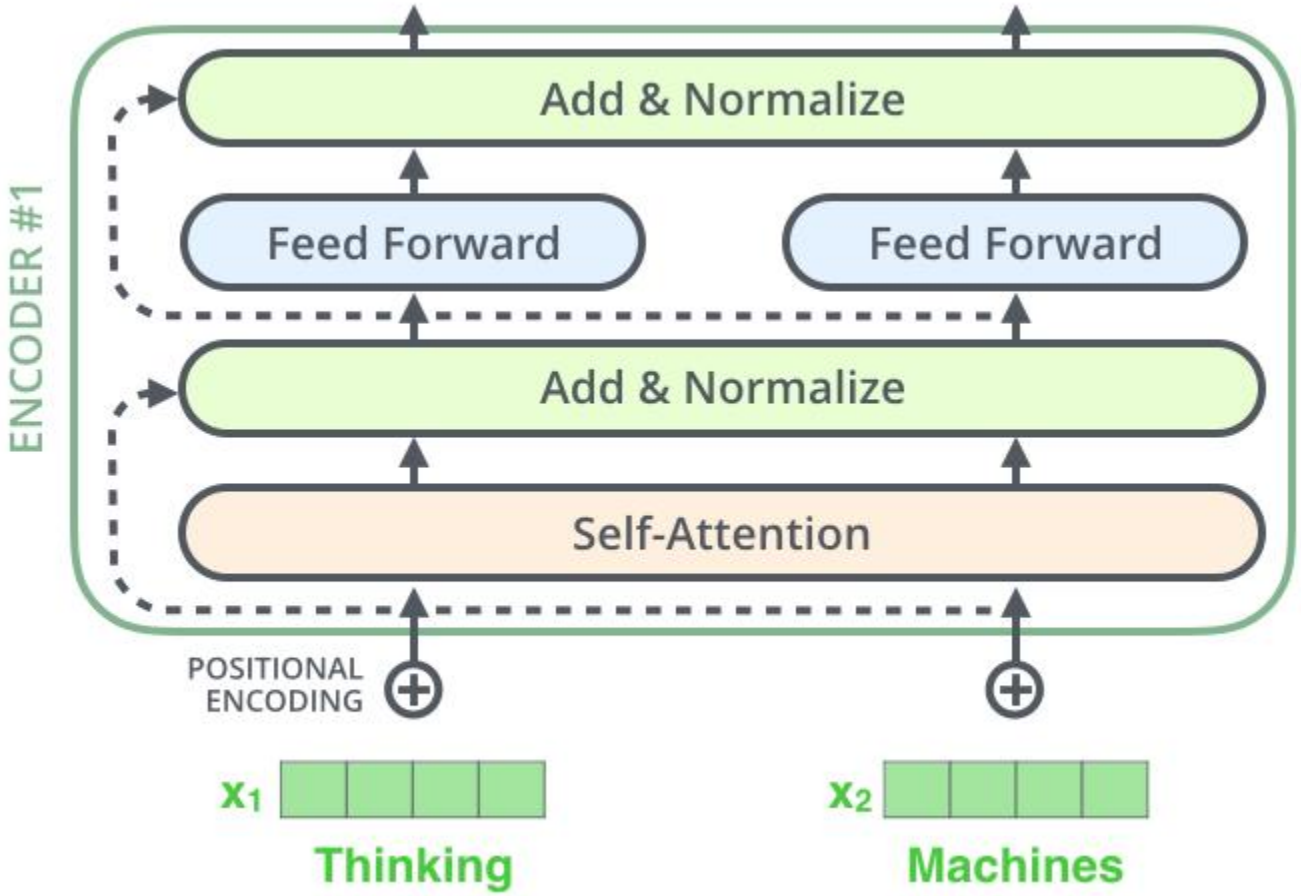
Transformer
Positional Encoding

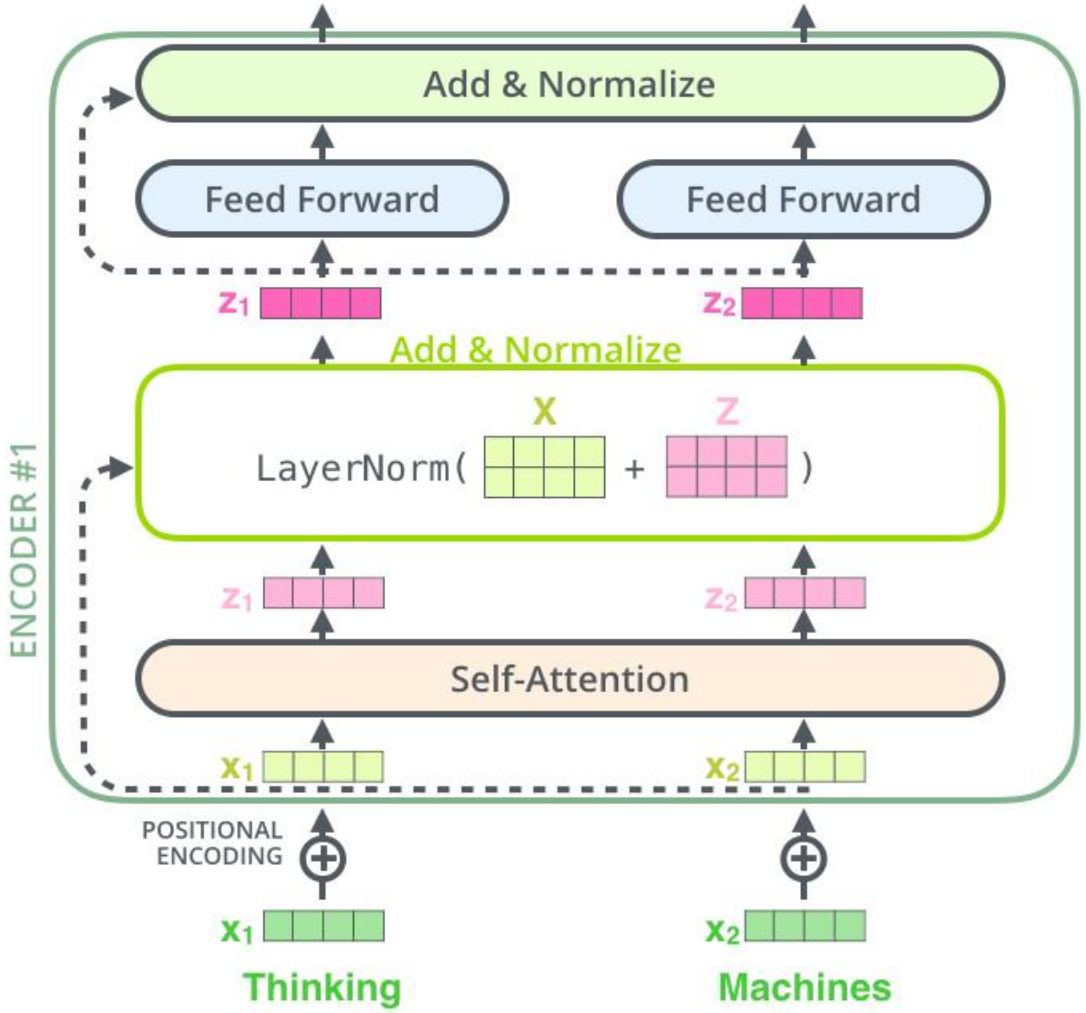


Transformer
Positional Encoding

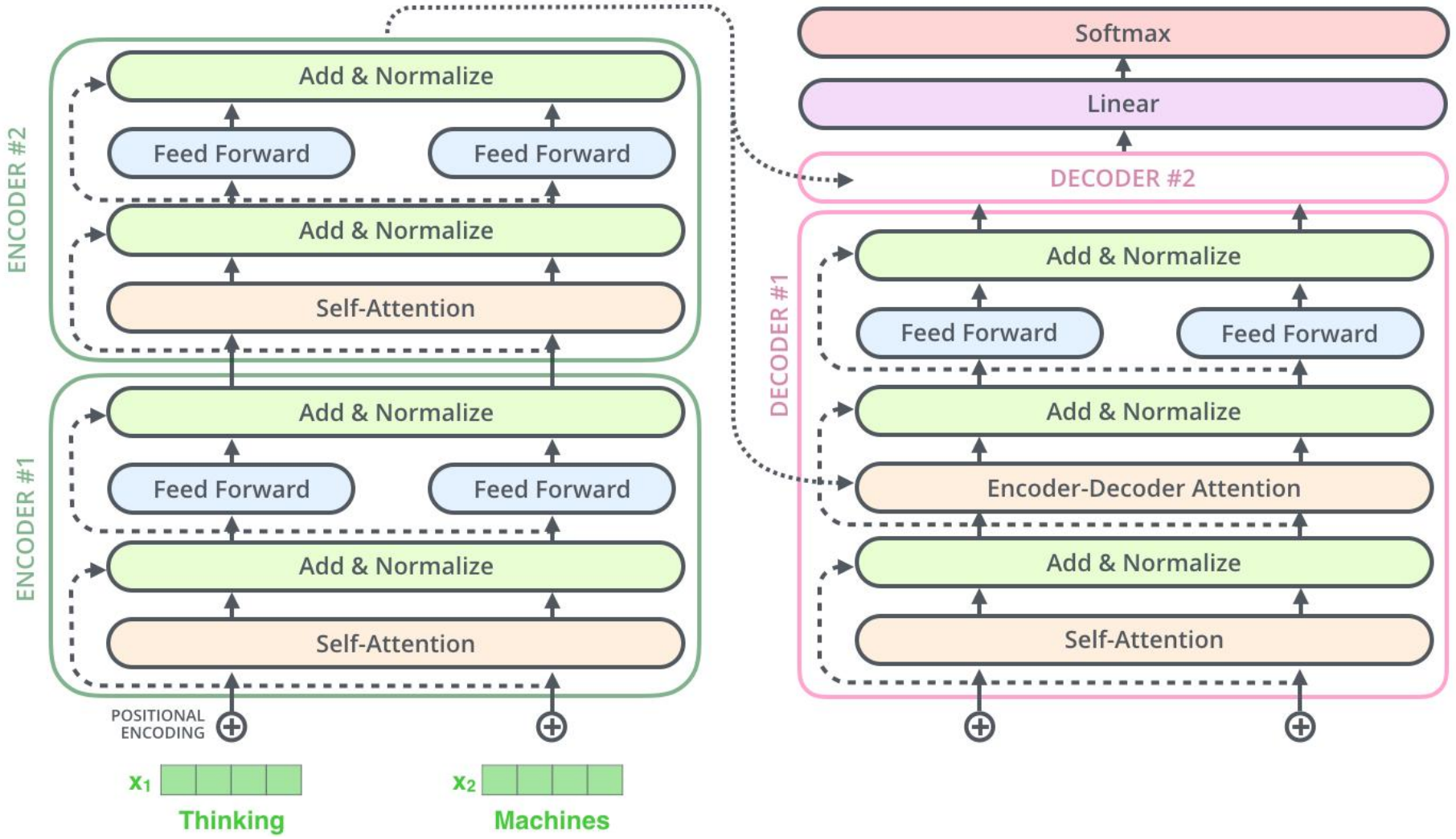


Transformer
LayerNorm&Residuals





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。

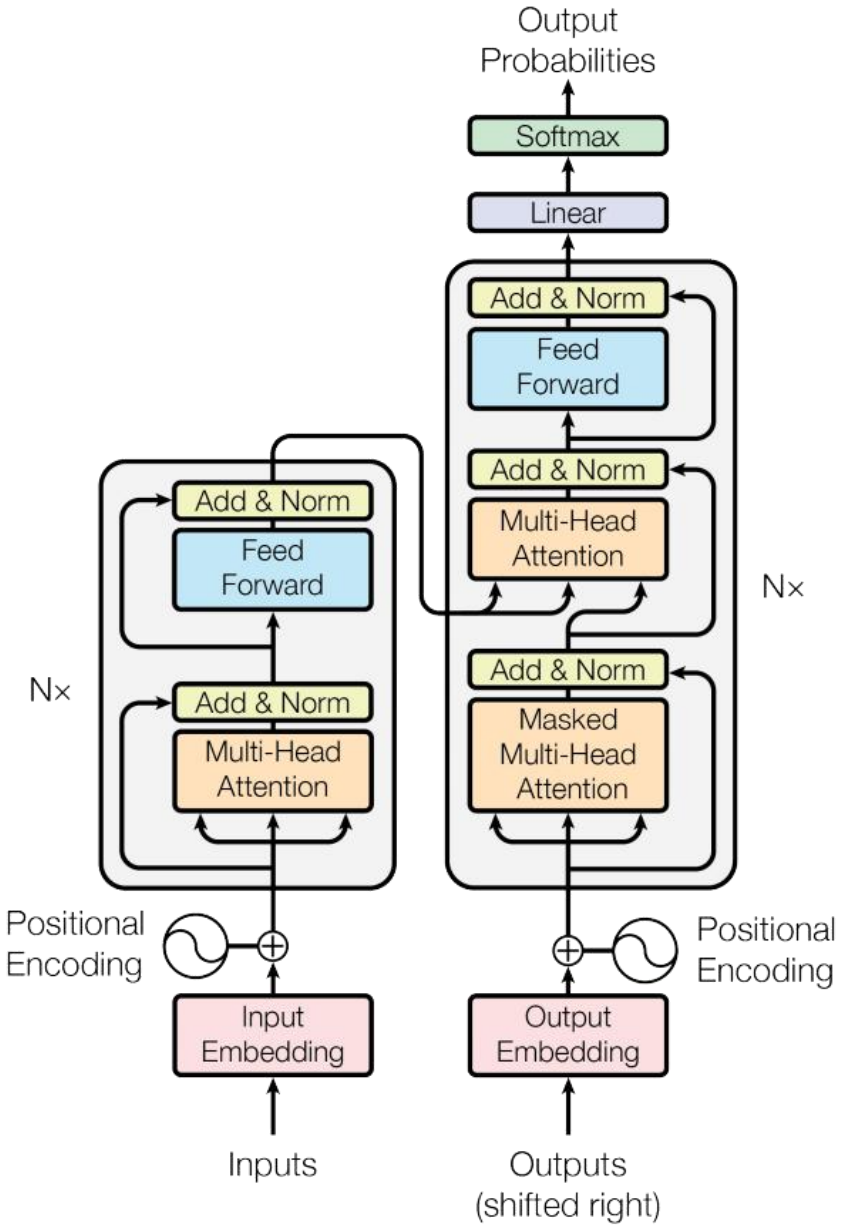
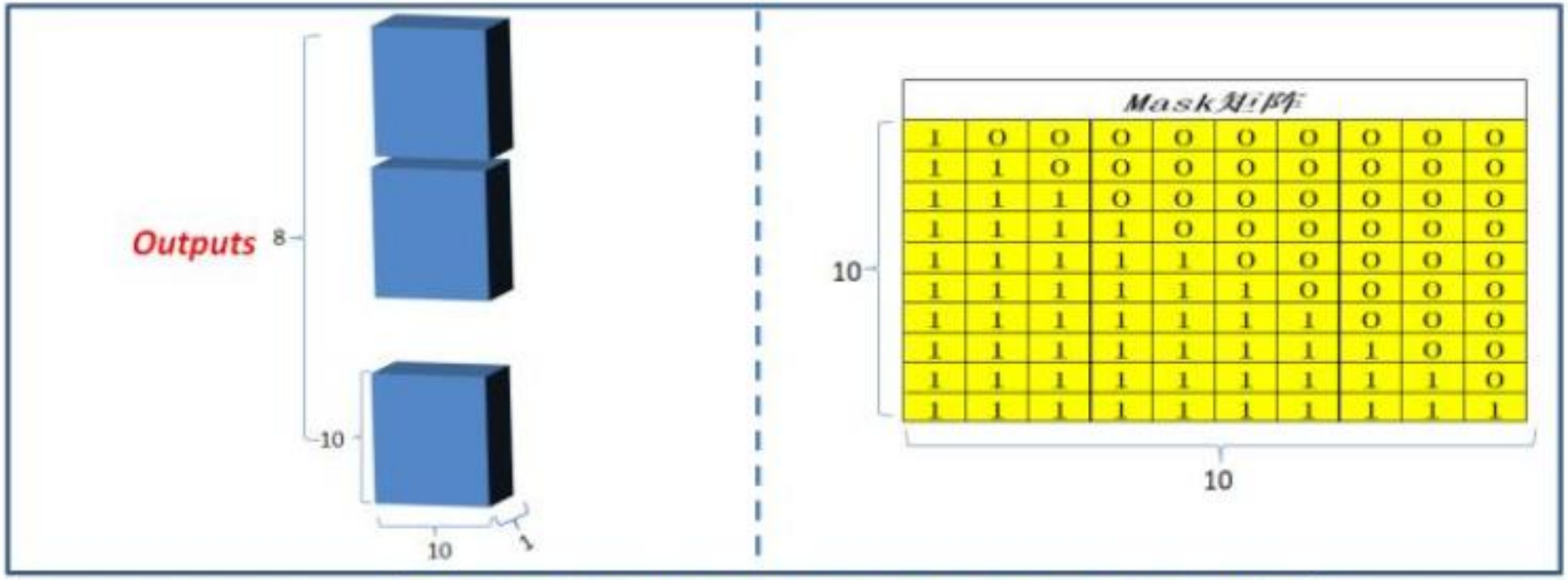
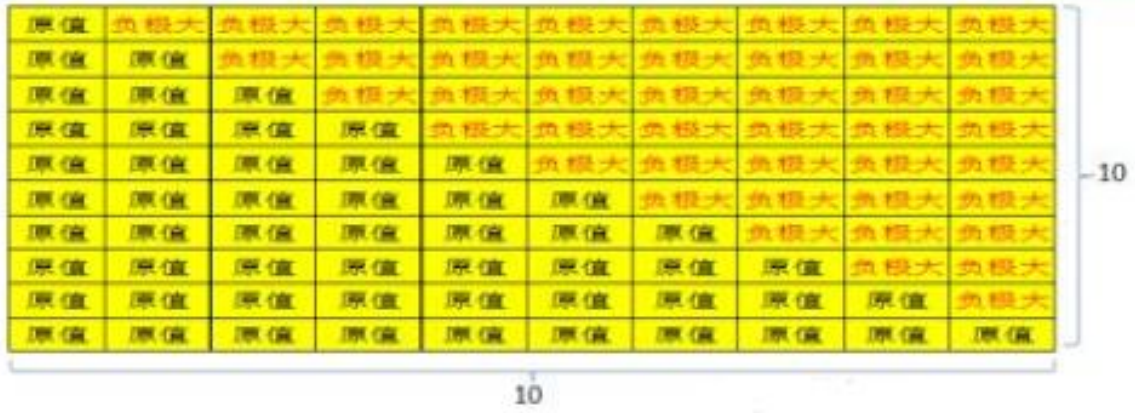


Figure 1: The Transformer - model architecture.

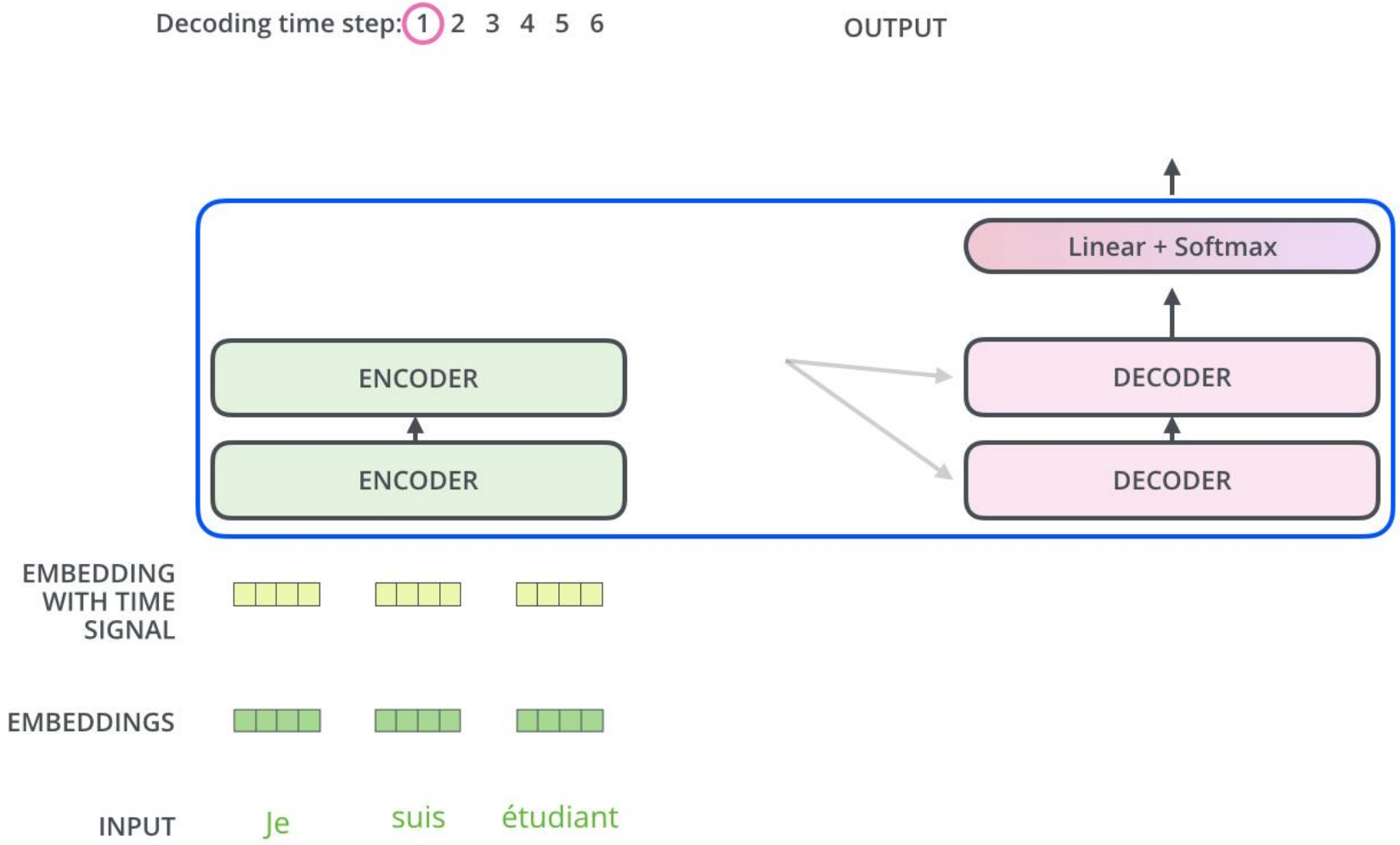
Transformer
Decoder Masked Multi-Headed-Attention



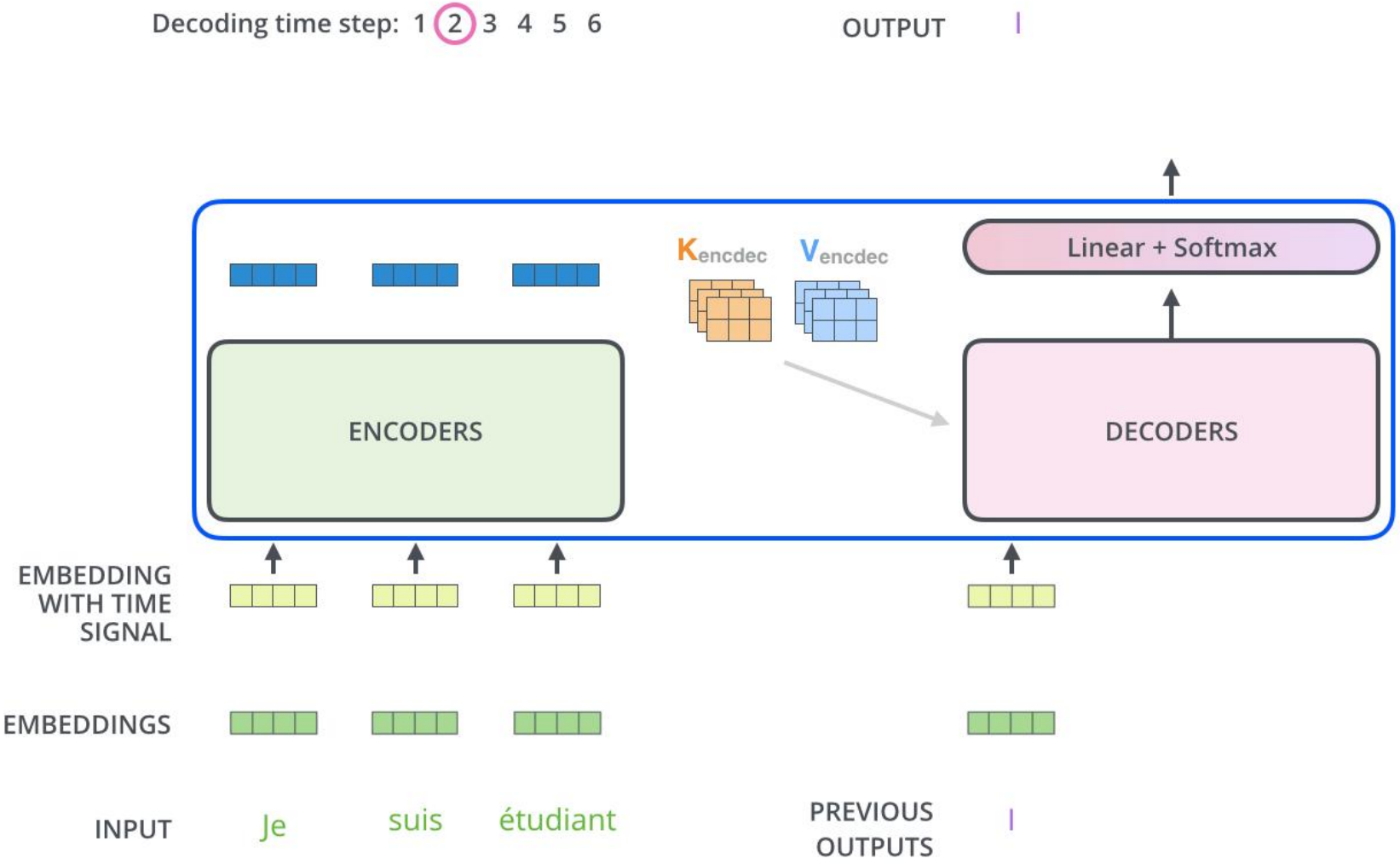
用Mask矩阵作用于Outputs中的每一个10X10单元矩阵：
mask矩阵中元素‘1’对应的Outputs单元元素保留原值，
‘0’对应的Outputs单元元素替换为负极大值：



Transformer
Decoder



Transformer
Decoder



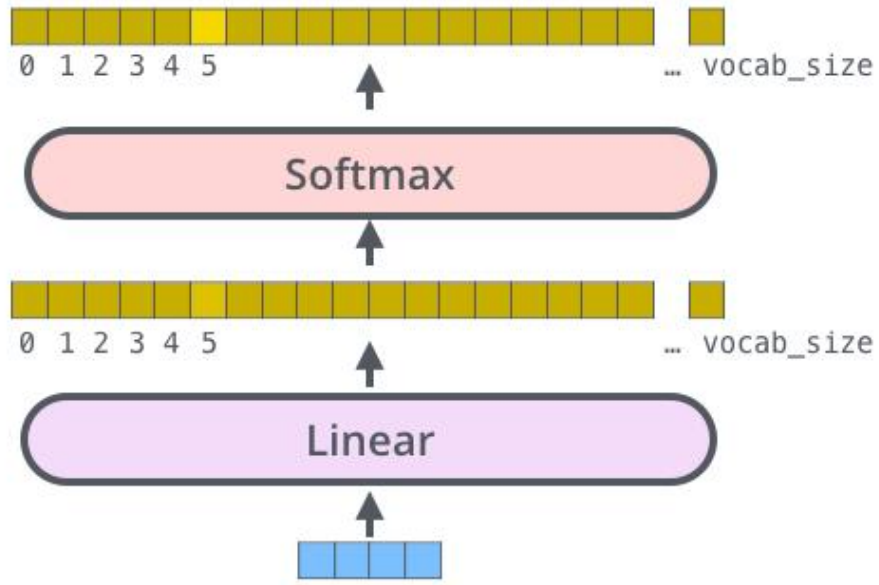
Transformer
Final Linear and Softmax Layer

Which word in our vocabulary is associated with this index?

am

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

5

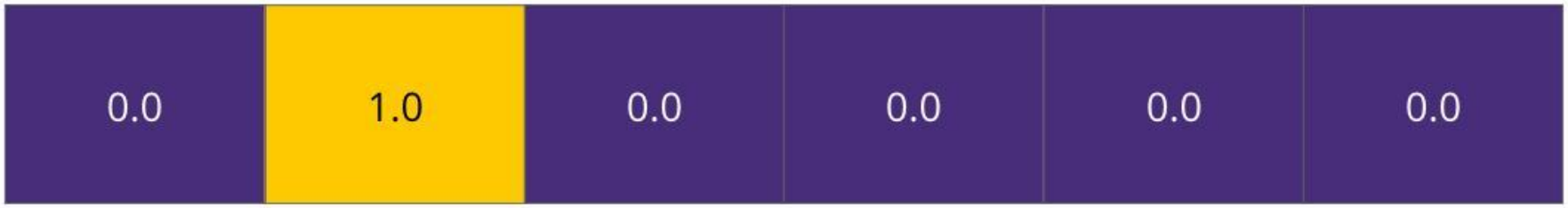


Transformer
Training

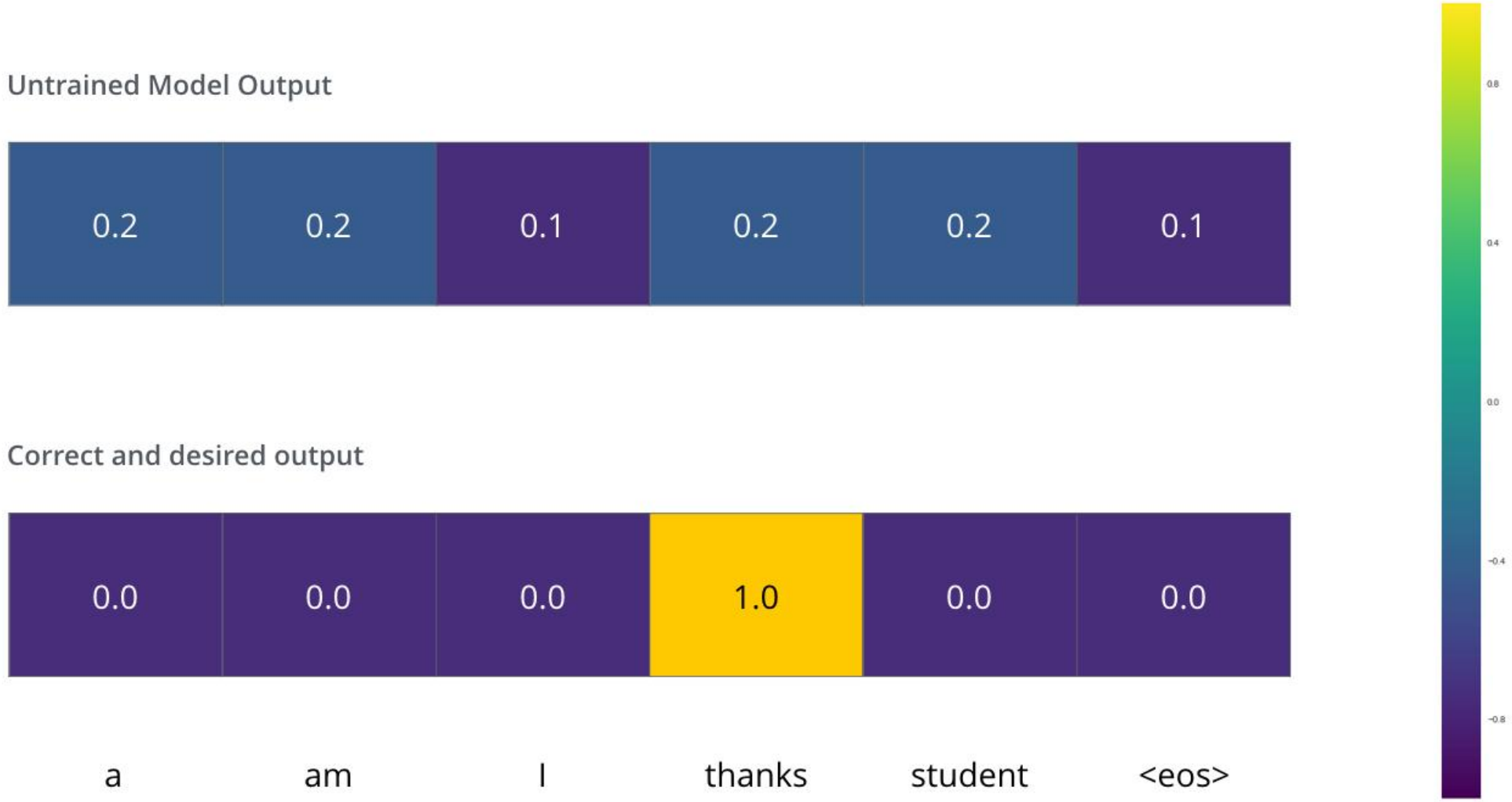
Output Vocabulary

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

One-hot encoding of the word “am”

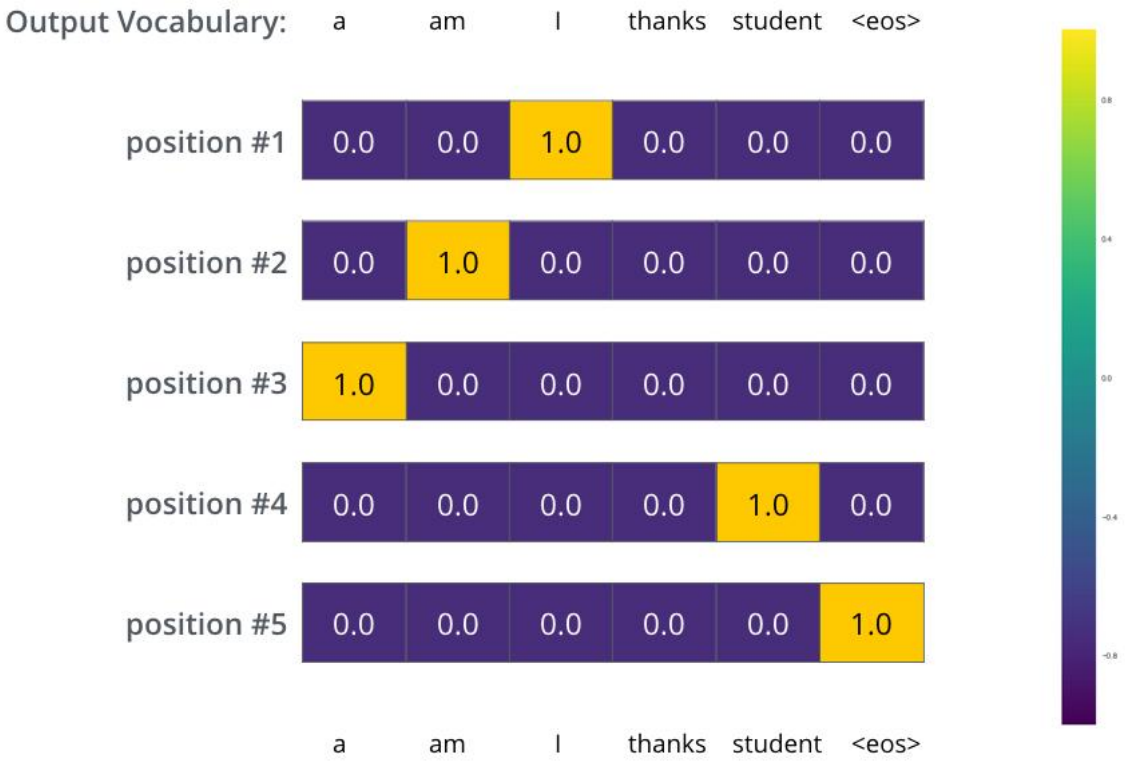


Transformer
Training

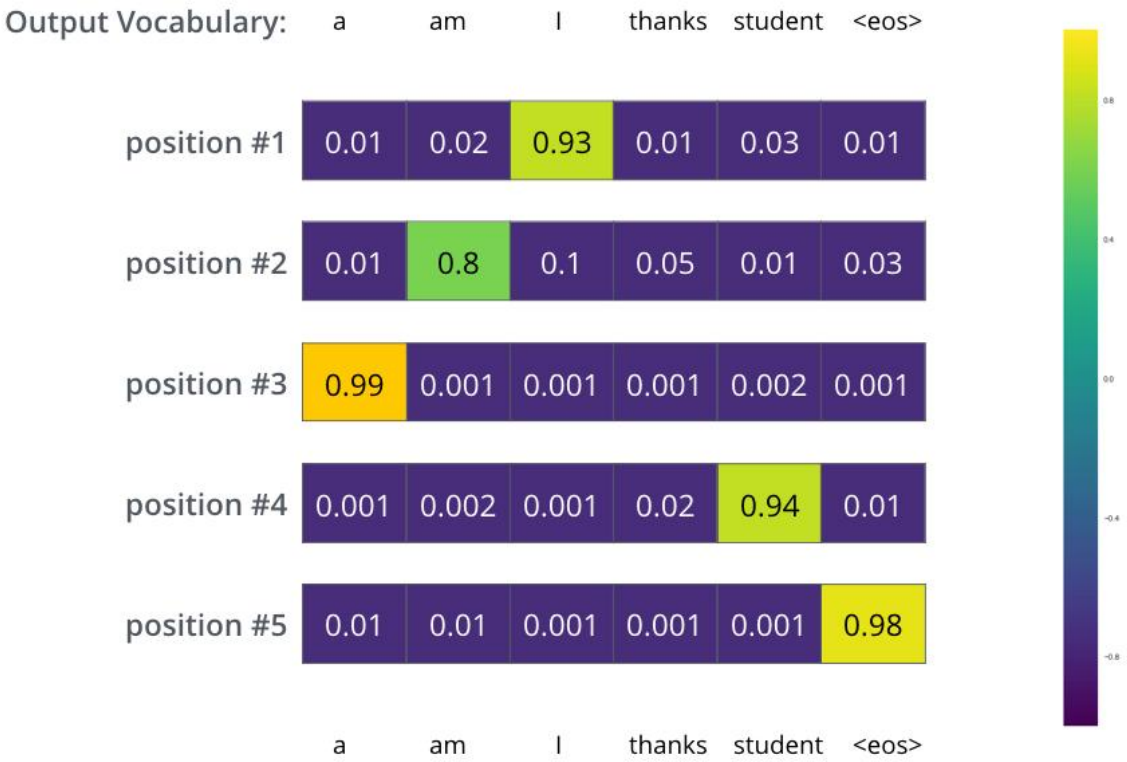


Transformer
Training

Target Model Outputs



Trained Model Outputs



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

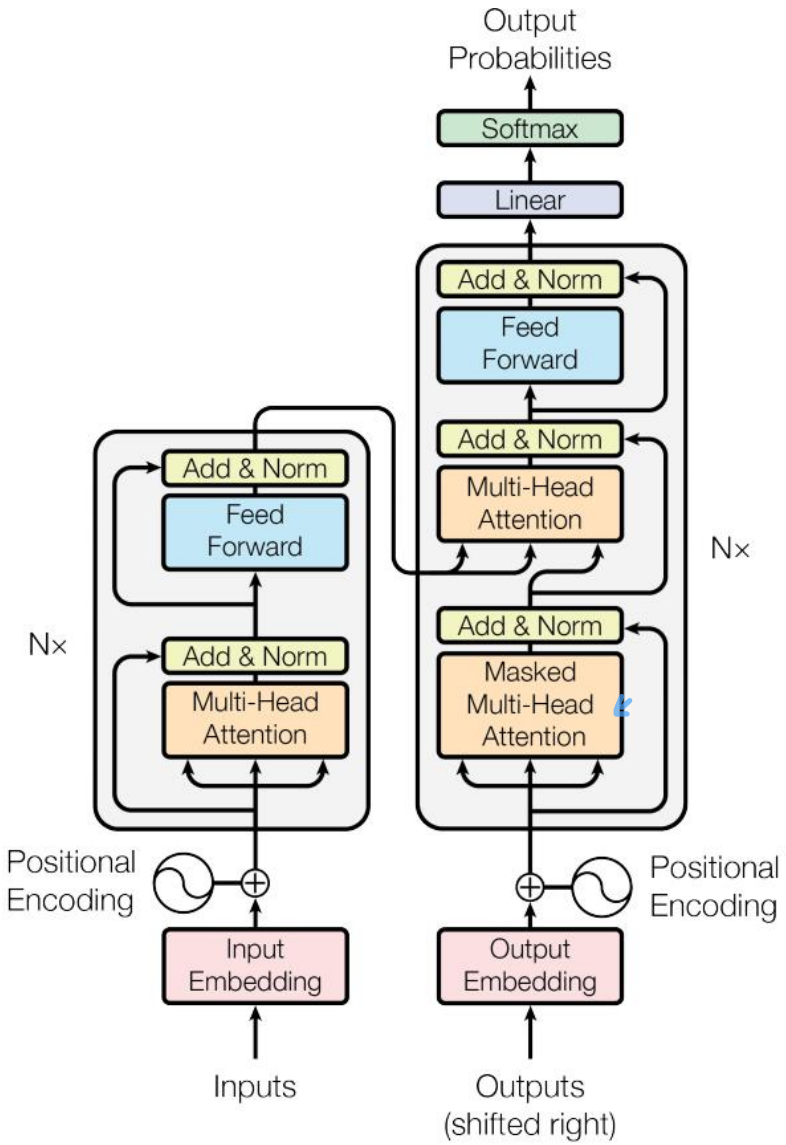
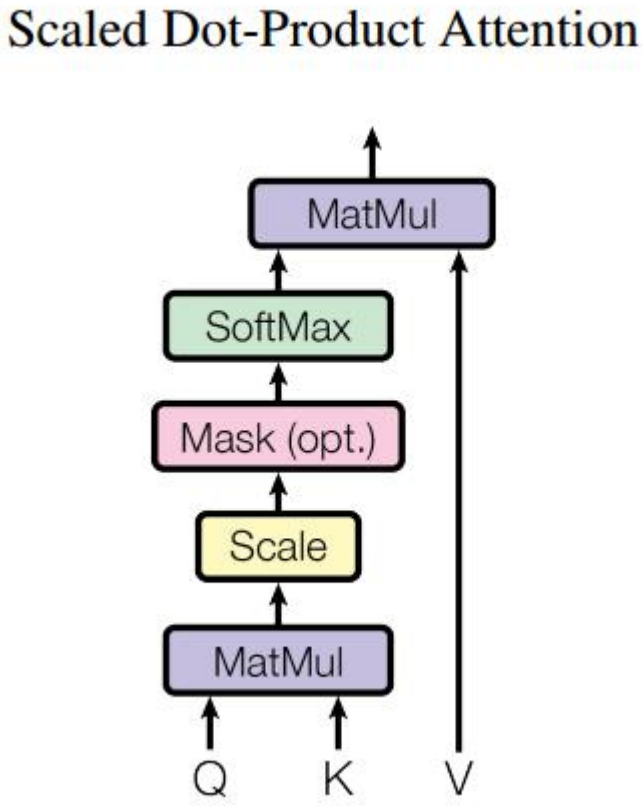


Figure 1: The Transformer - model architecture.

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。

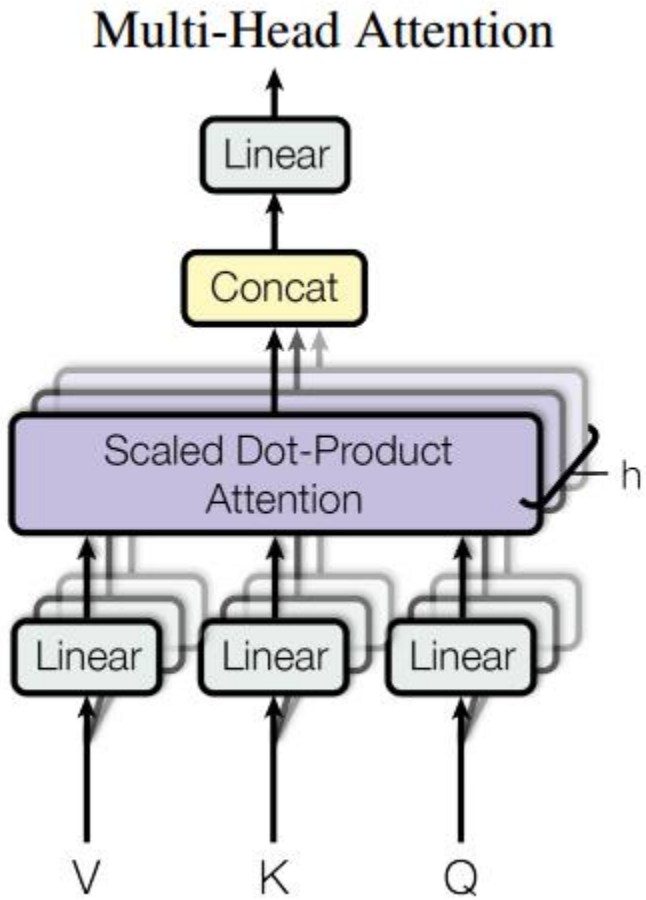
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



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。

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$



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!