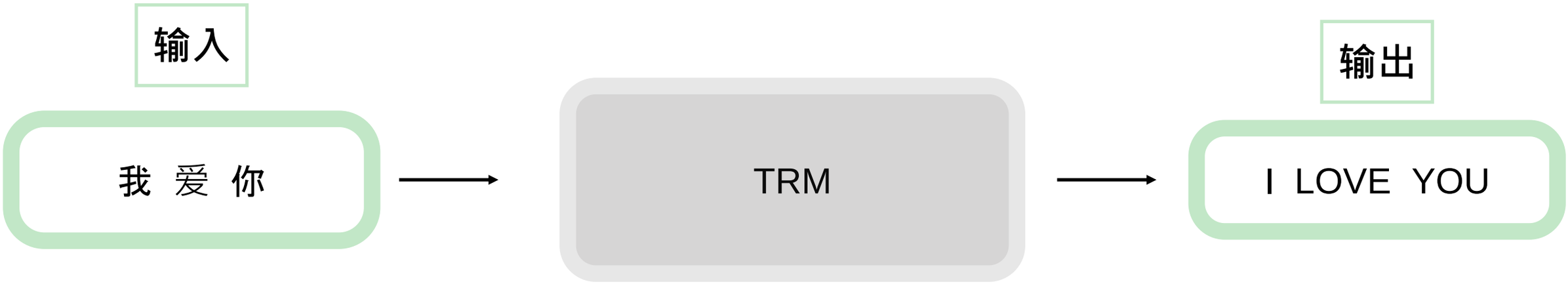
# Transformer 从零解读



**后台回复【答案解析】**

扫码关注微信公众号

文章周更

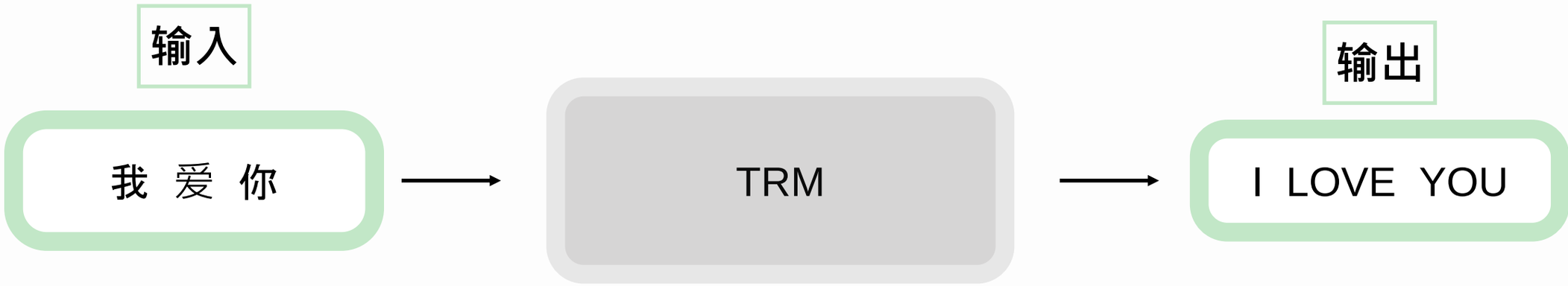
知识分享

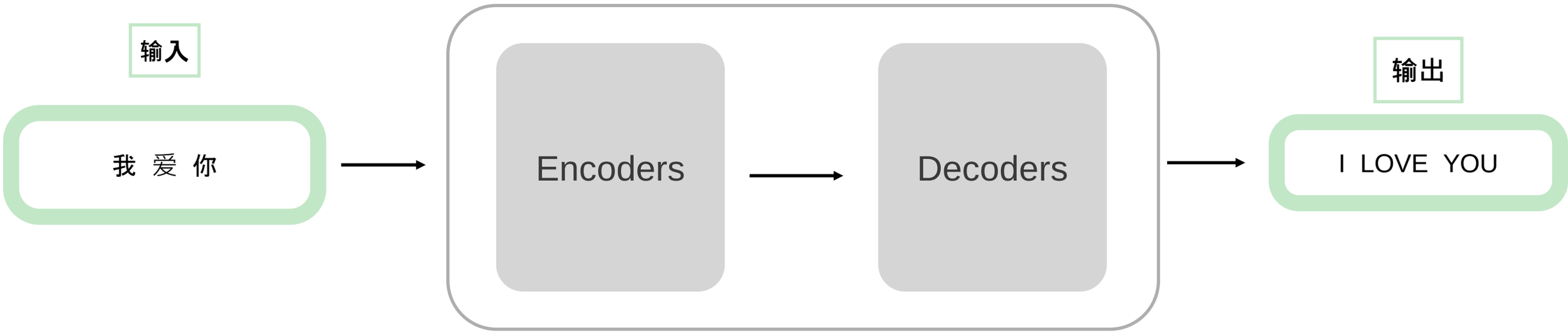
一起进步

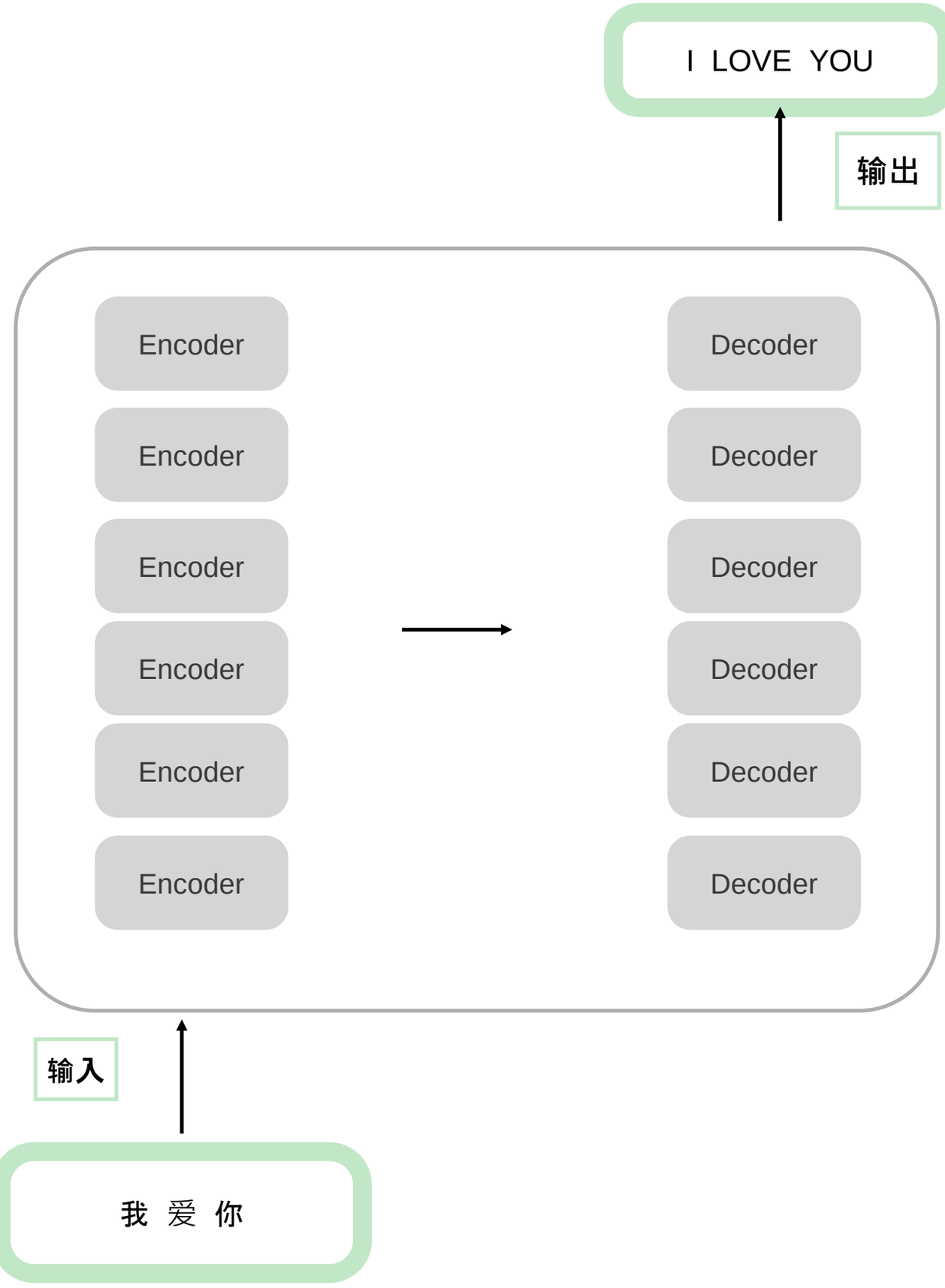
求关注，求点赞，求一切！！

1. 位置编码
2. 多头注意力机制
3. 残差和layerNorm
4. 前馈神经网络
5. TRM面试题讲解

## TRM在做一个什么事情？




输入
我 爱 你
Encoders
Decoders
输出
I LOVE YOU


I LOVE YOU
输出
Encoder
Encoder
Encoder
Encoder
Encoder
Encoder
Encoder
Decoder
Decoder
Decoder
Decoder
Decoder
Decoder
Decoder
输入
我 爱 你

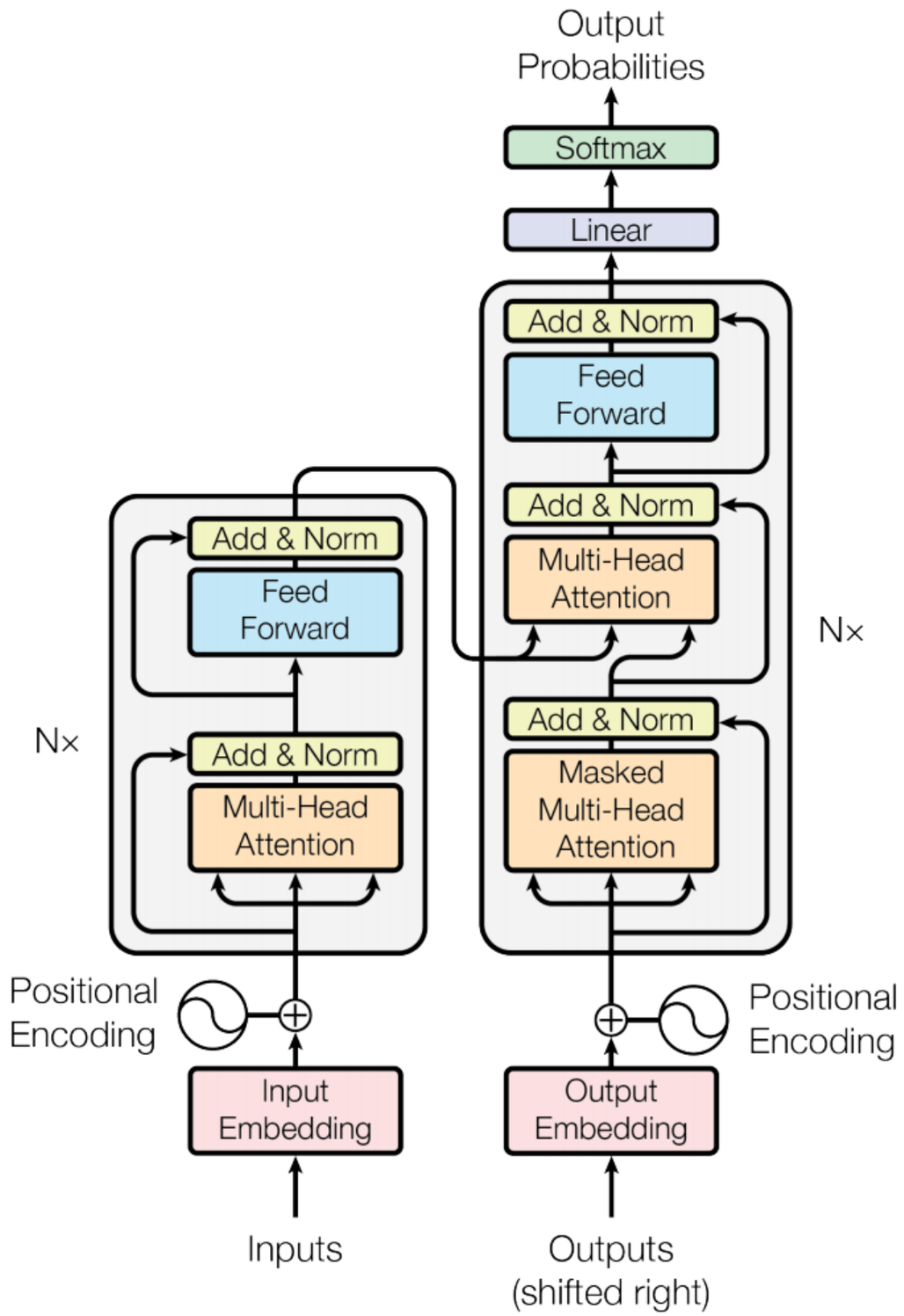
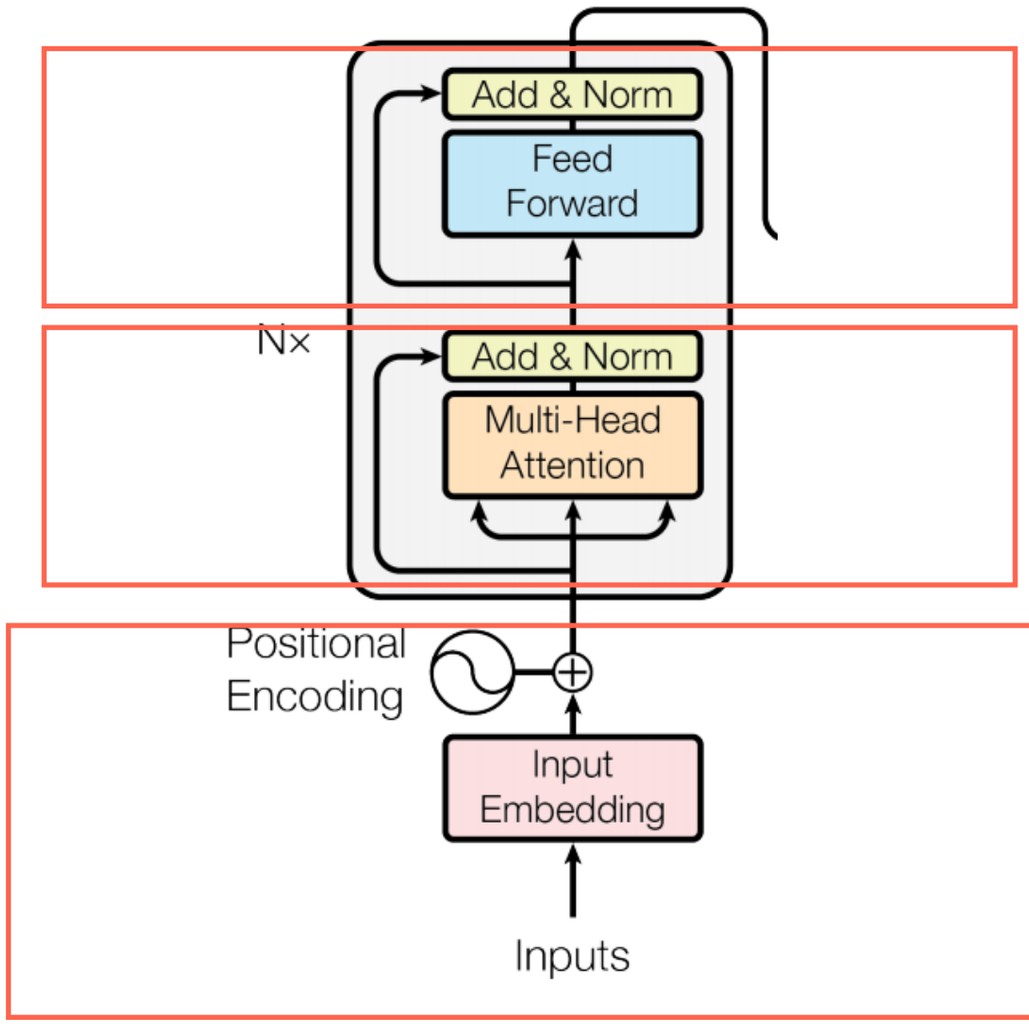



Figure 1: The Transformer - model architecture.




3 前馈神经网络

2 注意力机制

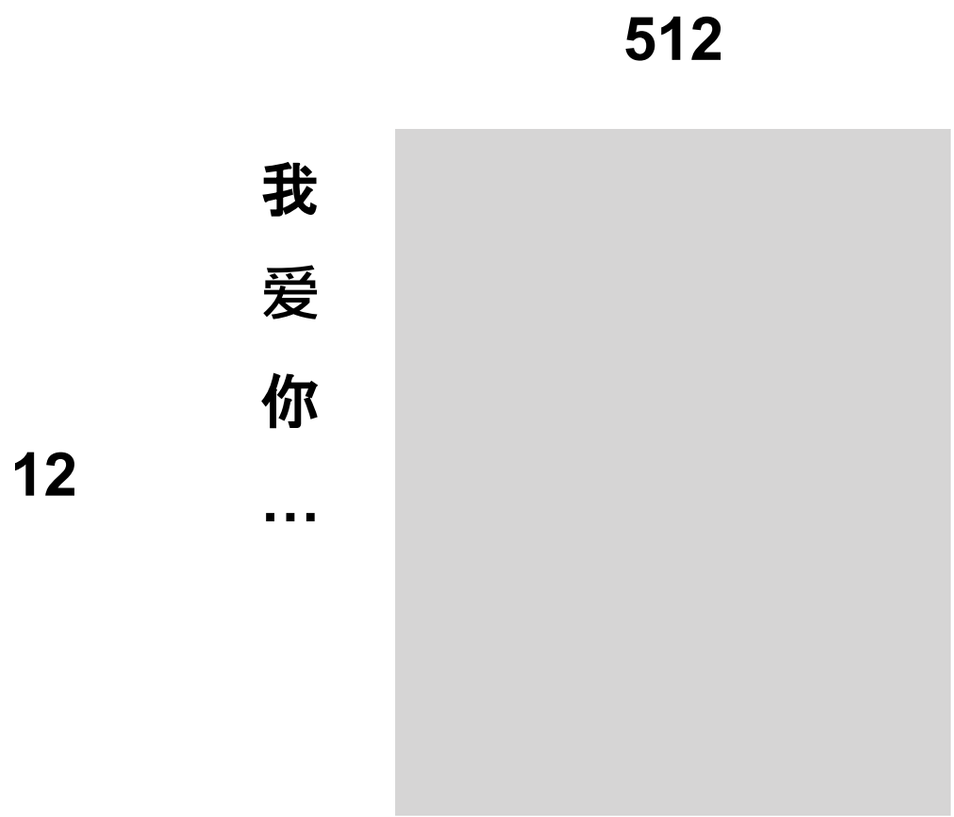
1 输入部分

输入部分

**1. Embedding**

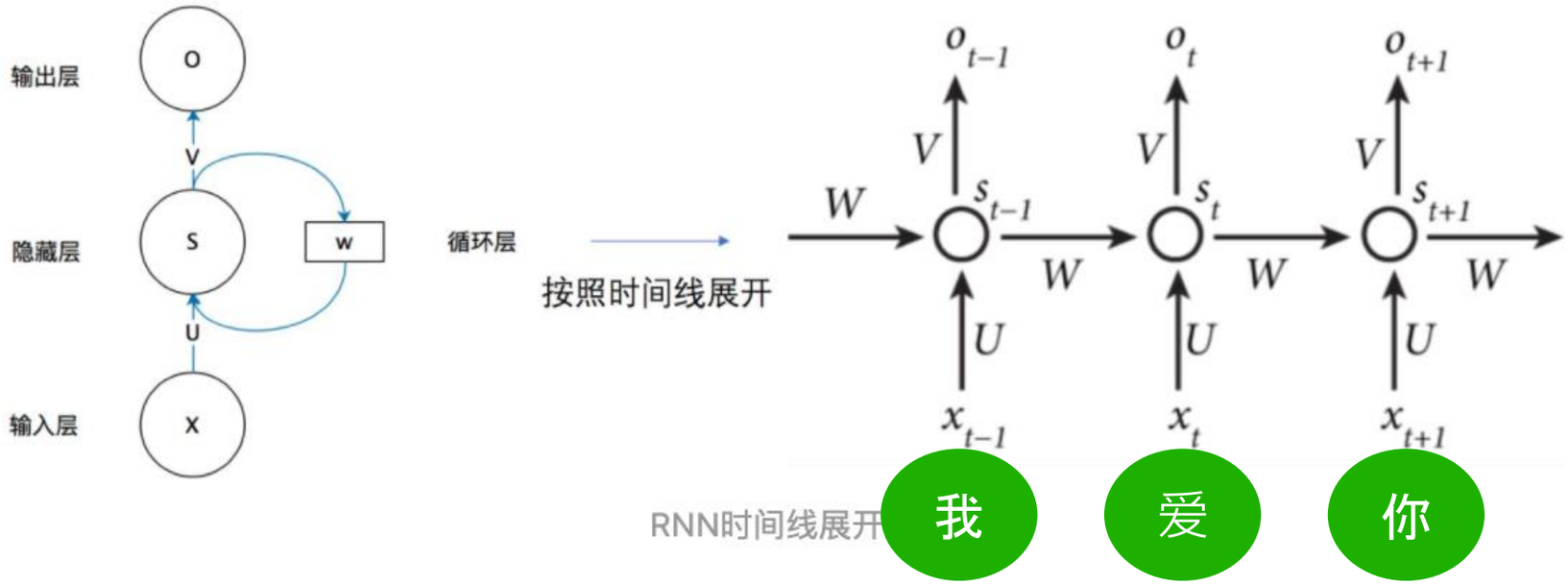
**2. 位置嵌入**

# Embedding



## 位置编码

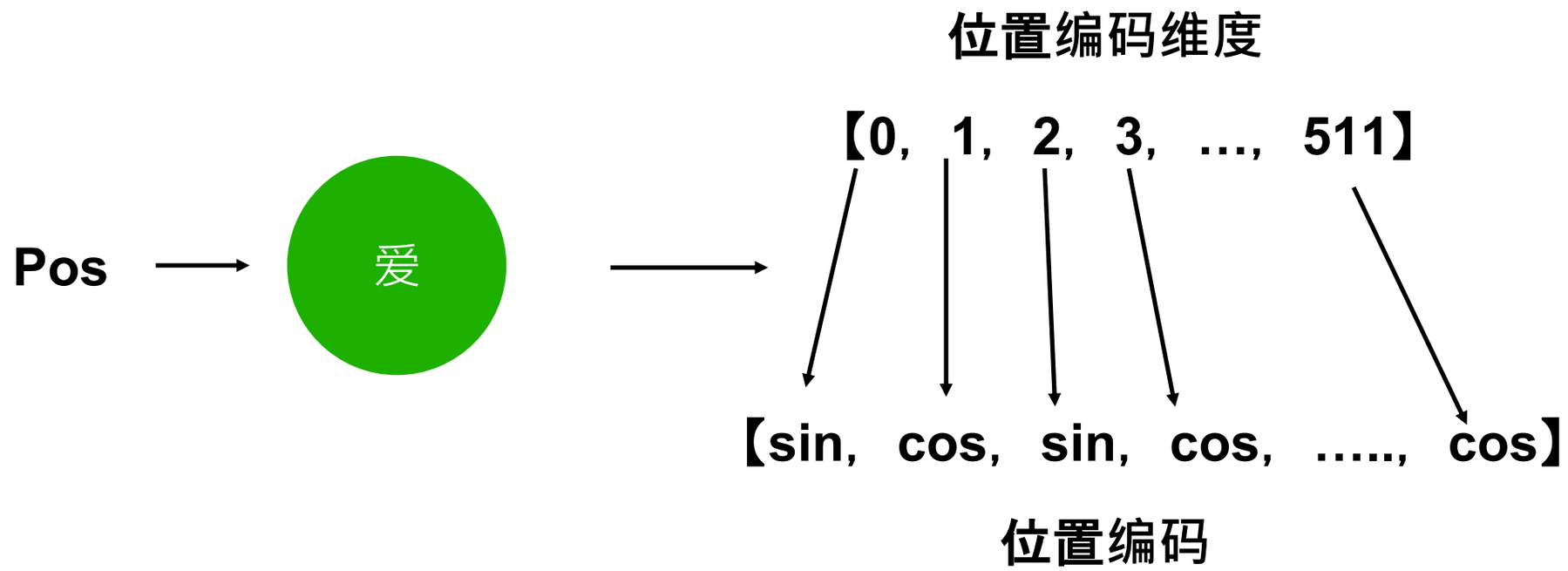
为什么需要：

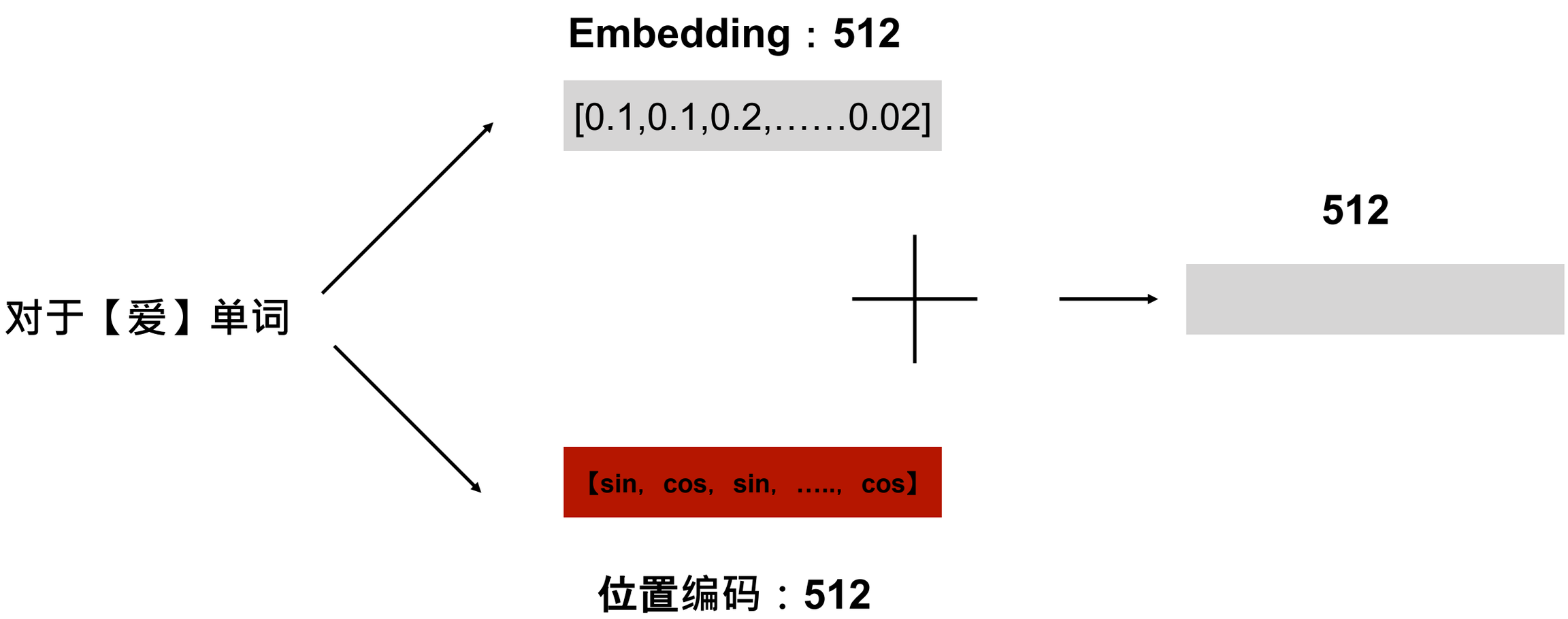



RNN时间线展开

## 位置编码公式

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




Embedding：512
[0.1,0.1,0.2,……0.02]
512
对于【爱】单词
【sin，cos，sin，…..，cos】
位置编码：512

## 引申一下为什么位置嵌入会有用

借助上述公式，我们可以得到一个特定位置的 $d_{model}$ 维的位置向量，并且借助三角函数的性质

$$\begin{cases} sin(\alpha+\beta) = sin\alpha cos\beta + cos\alpha sin\beta \\ cos(\alpha+\beta) = cos\alpha cos\beta - sin\alpha sin\beta \end{cases} \tag{2}$$

我们可以得到：

$$\begin{cases} PE(pos+k, 2i) = PE(pos, 2i) \times PE(k, 2i+1) + PE(pos, 2i+1) \times PE(k, 2i) \\ PE(pos+k, 2i+1) = PE(pos, 2i+1) \times PE(k, 2i+1) - PE(pos, 2i) \times PE(k, 2i) \end{cases} \tag{3}$$

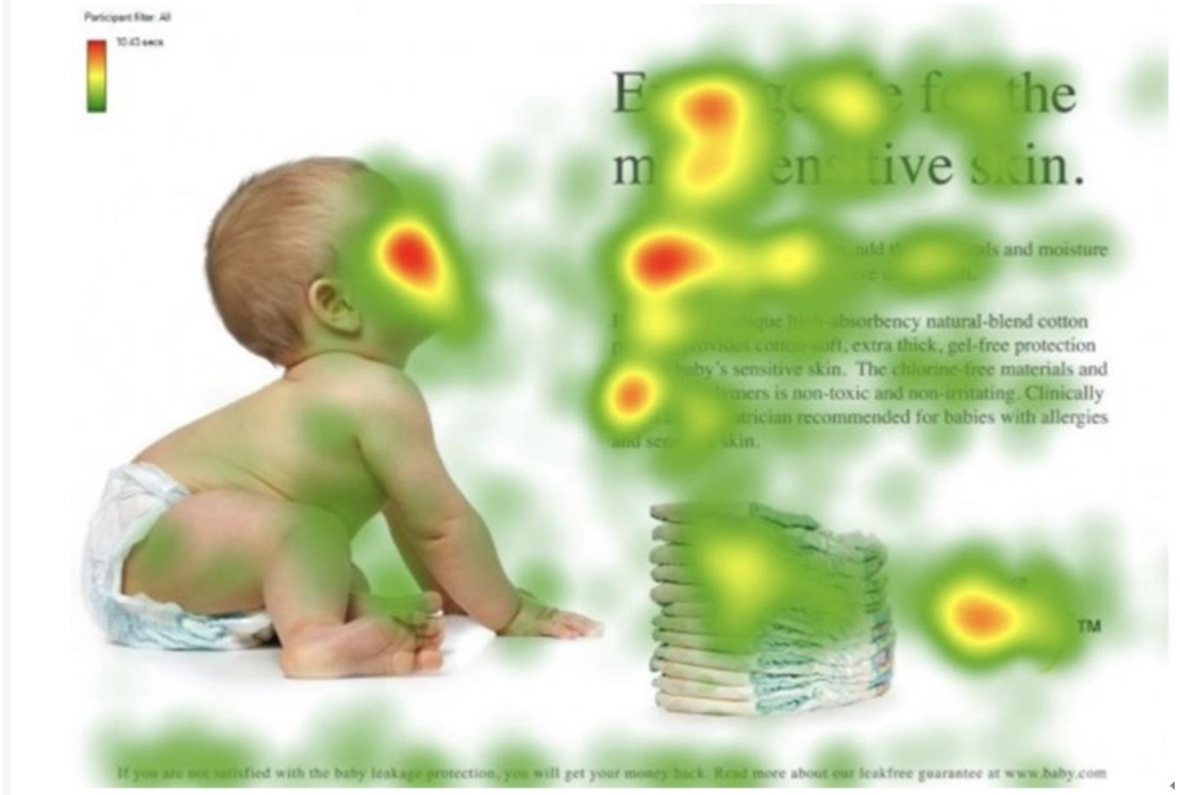
可以看出，对于 pos+k 位置的位置向量某一维 $2i$ 或 $2i+1$ 而言，可以表示为，pos 位置与 k 位置的位置向量的 $2i$ 与 $2i+1$ 维的线性组合，这样的线性组合意味着位置向量中蕴含了相对位置信息。

## 但是这种相对位置信息会在注意力机制那里消失

# 注意力机制

1. 基本的注意力机制
2. 在TRM中怎么操作

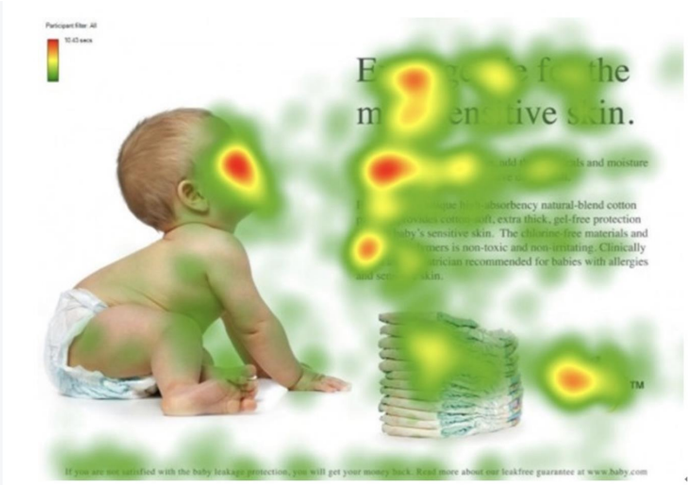
# 注意力机制本质

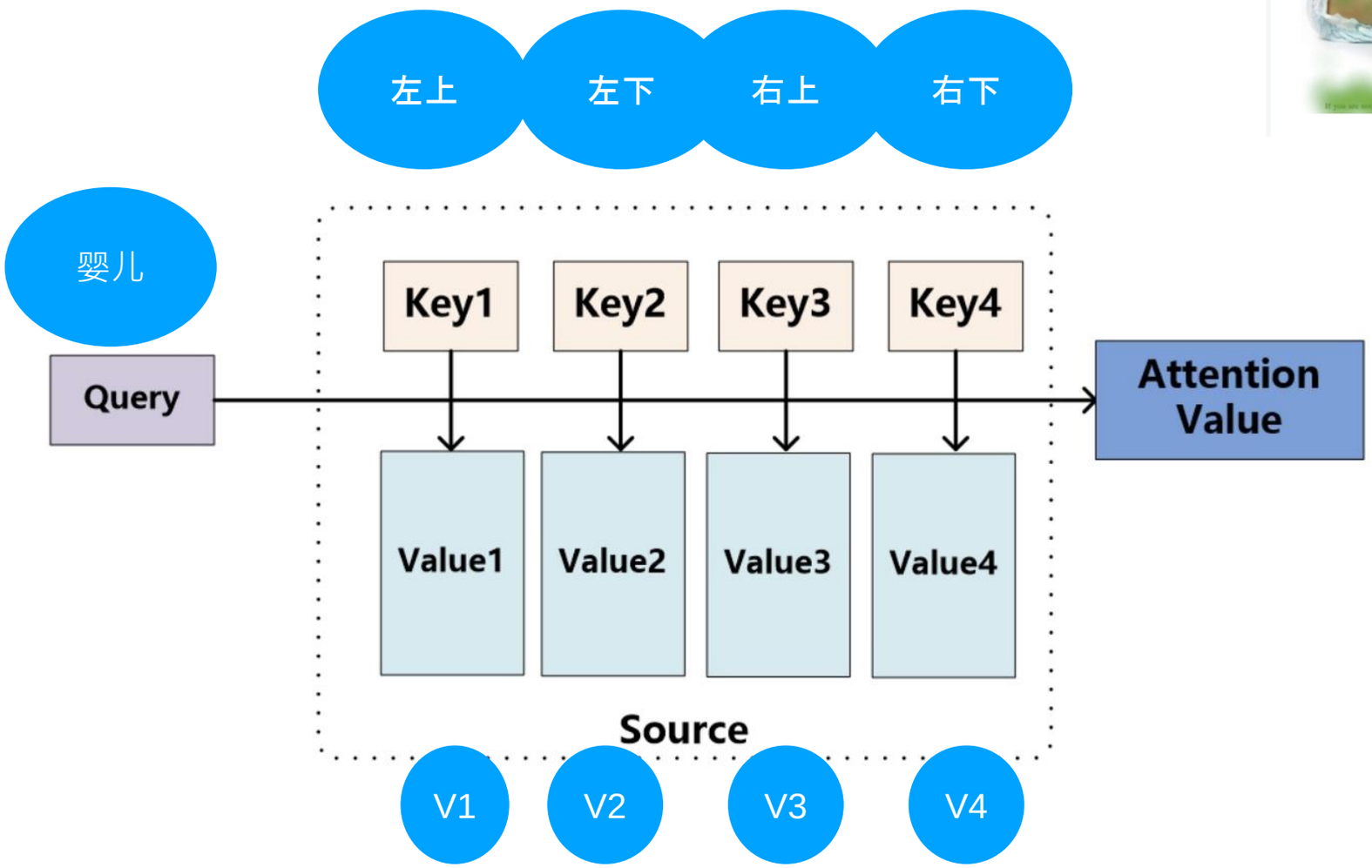



婴儿在干嘛

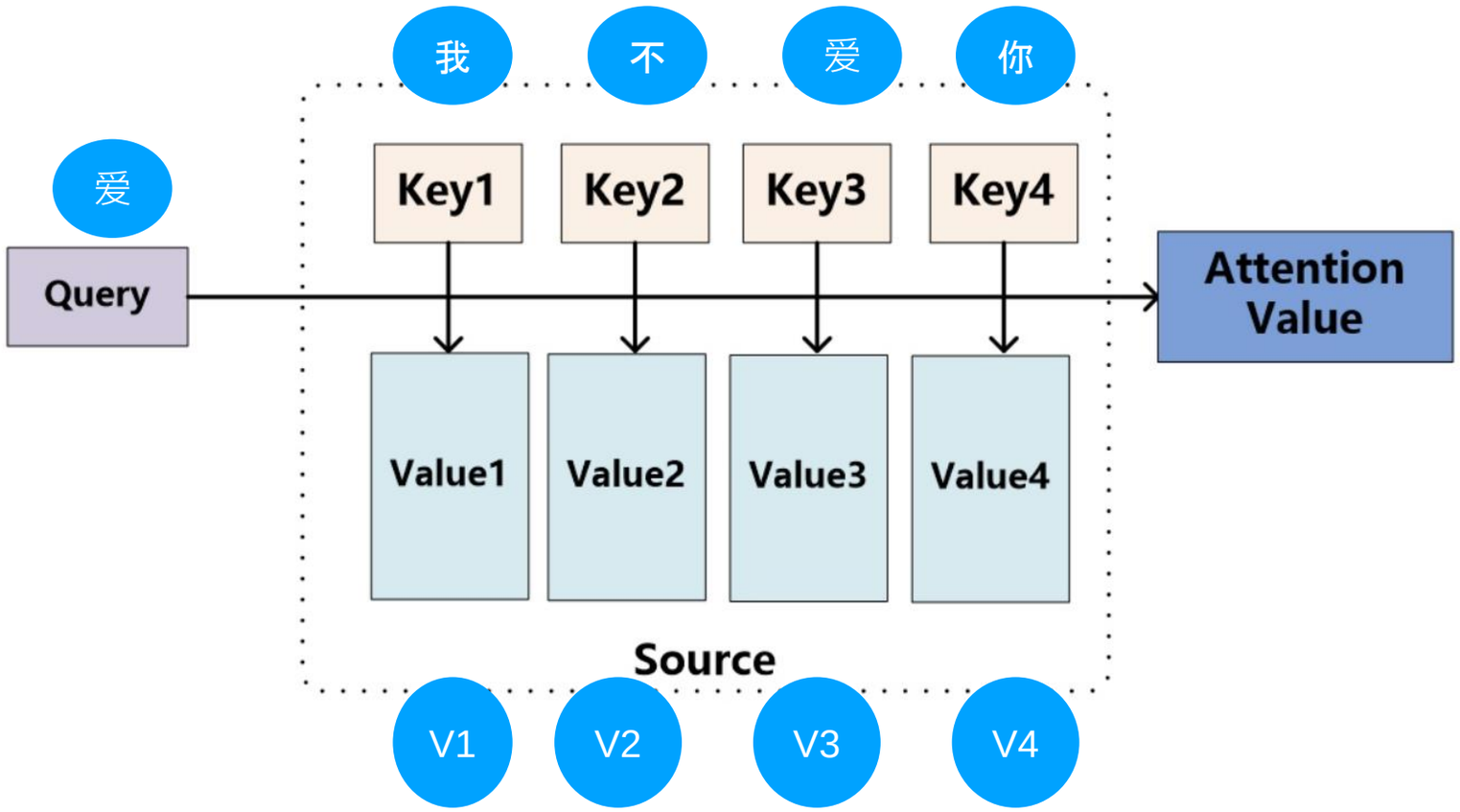
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

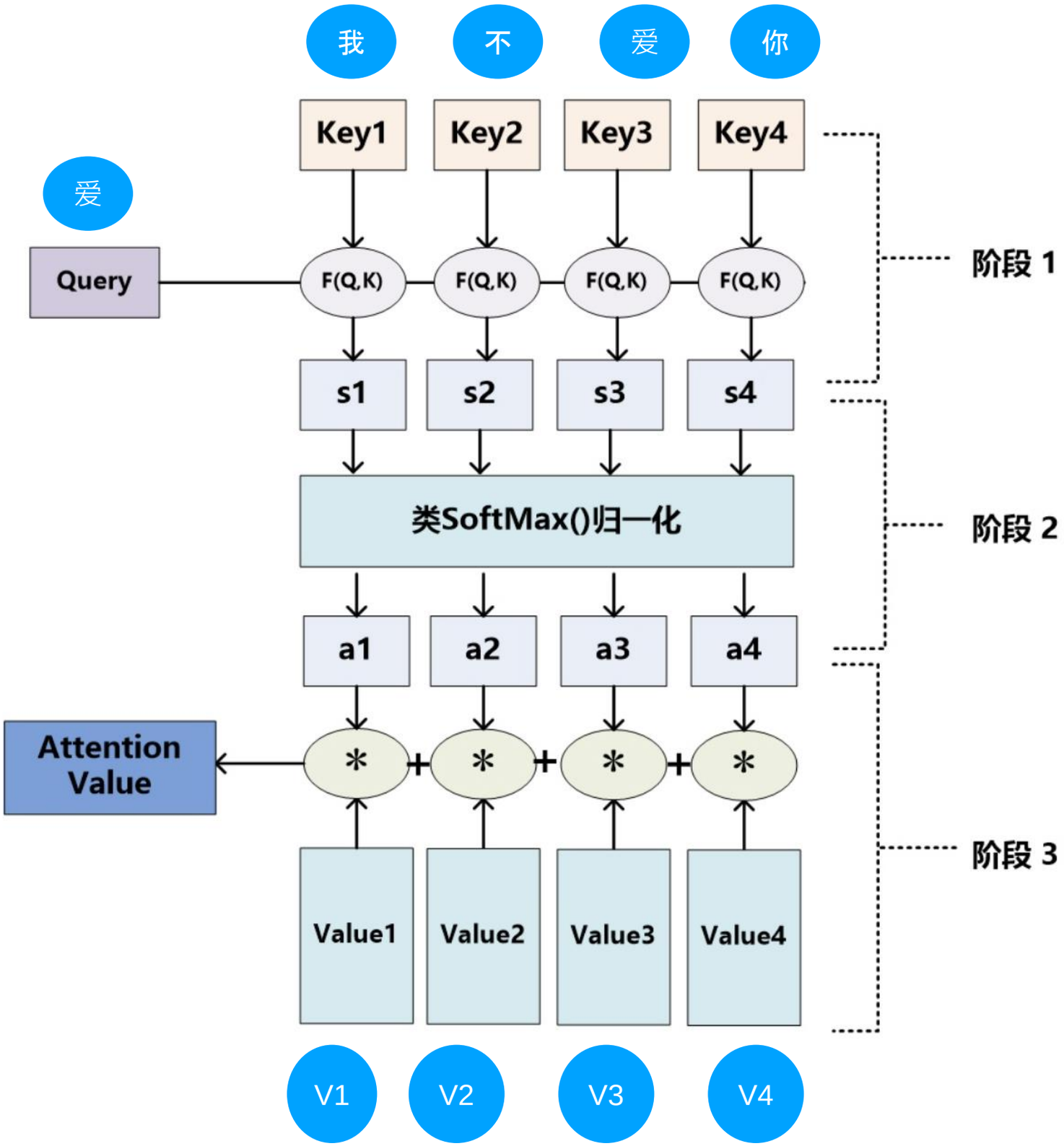
# 从公式角度来看：拿上面图片举例子





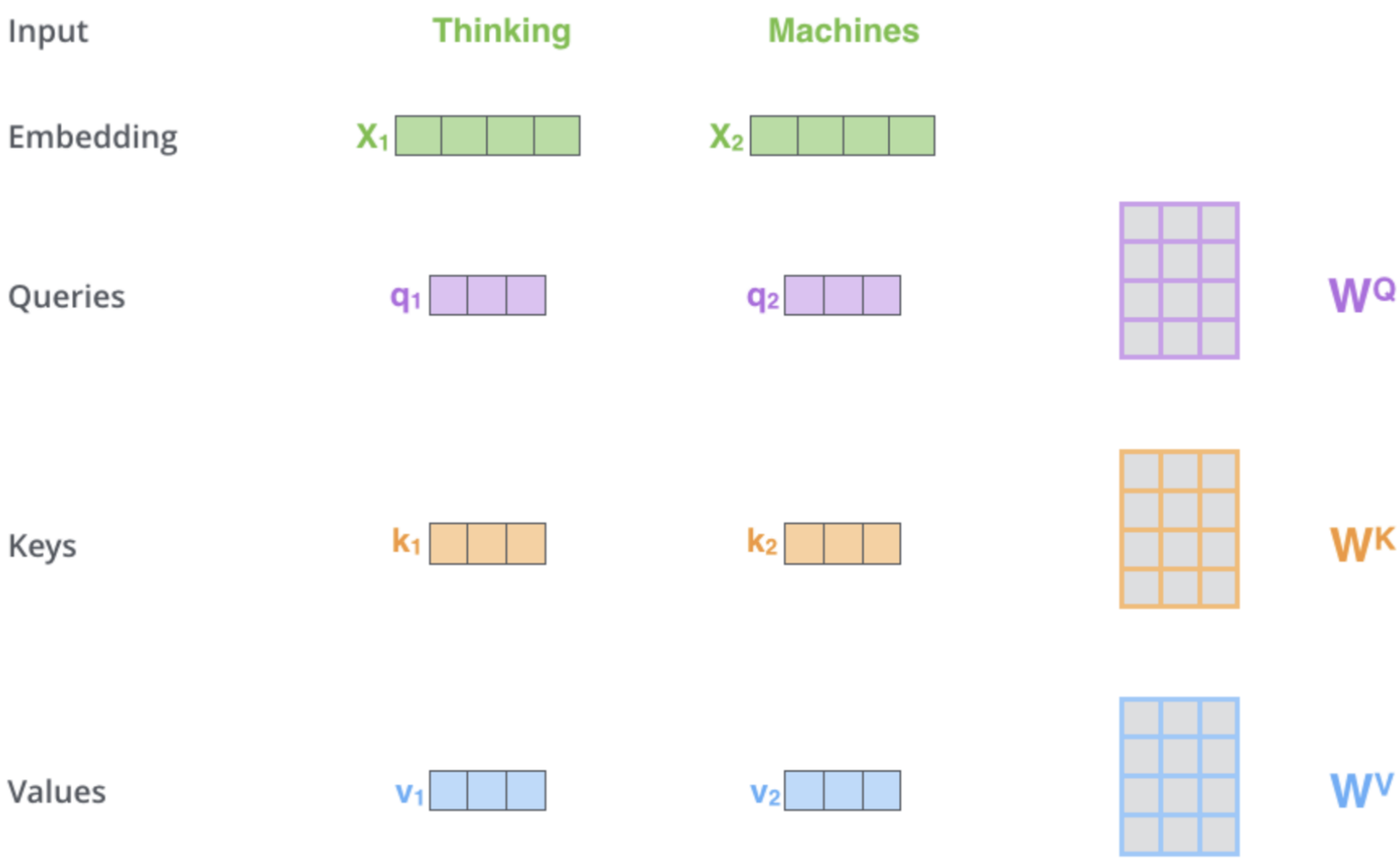
# 从公式角度来看


我
不
爱
你
爱
Key1
Key2
Key3
Key4
Query
Attention Value
Value1
Value2
Value3
Value4
Source
V1
V2
V3
V4

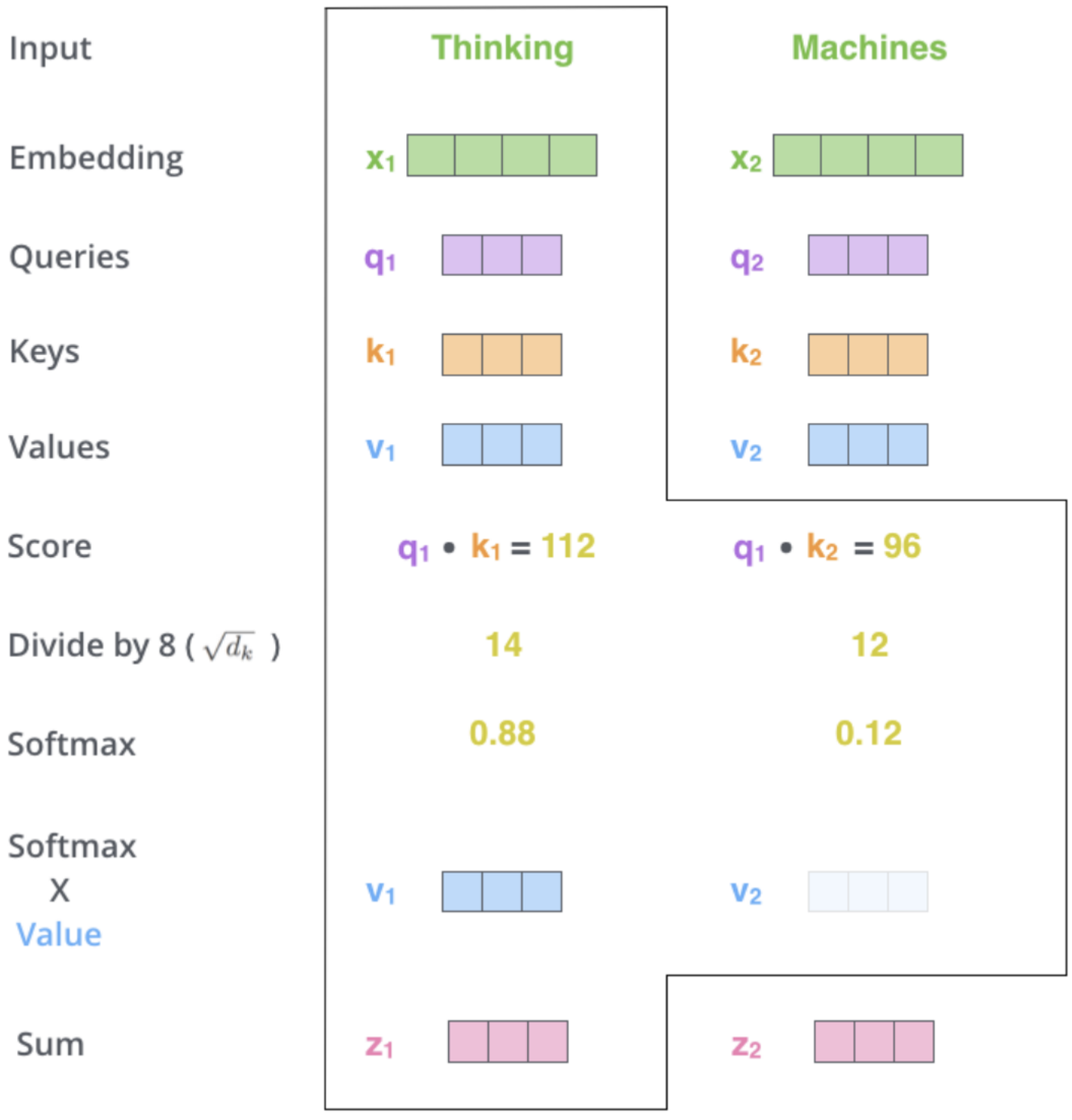

我
不
爱
你
Key1
Key2
Key3
Key4
爱
Query
F(Q,K)
F(Q,K)
F(Q,K)
F(Q,K)
阶段 1
s1
s2
s3
s4
类SoftMax()归一化
阶段 2
a1
a2
a3
a4
Attention Value
*
*
*
*
阶段 3
Value1
Value2
Value3
Value4
V1
V2
V3
V4

## TRM中的注意力

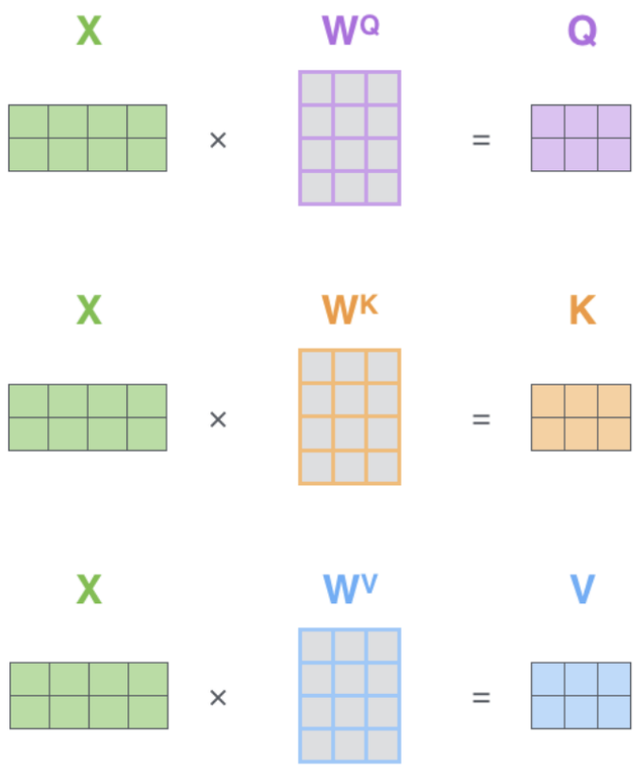
### 在只有单词向量的情况下，如何获取QKV

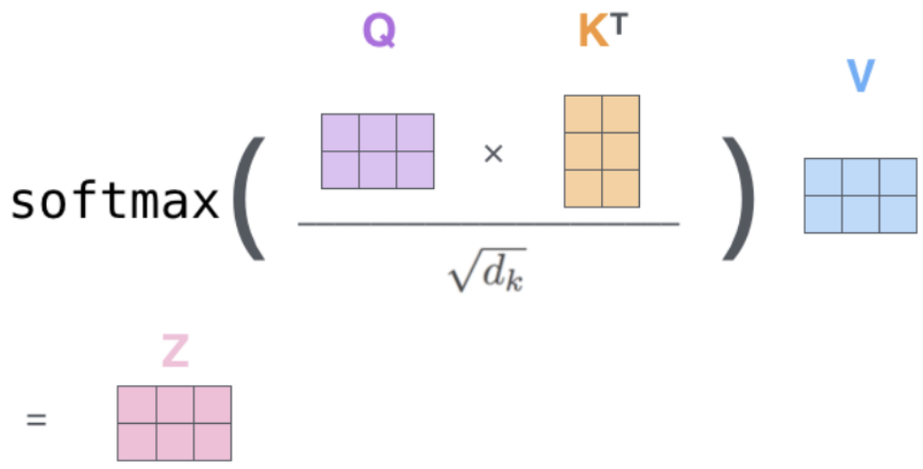


# 计算QK相似度，得到attention值



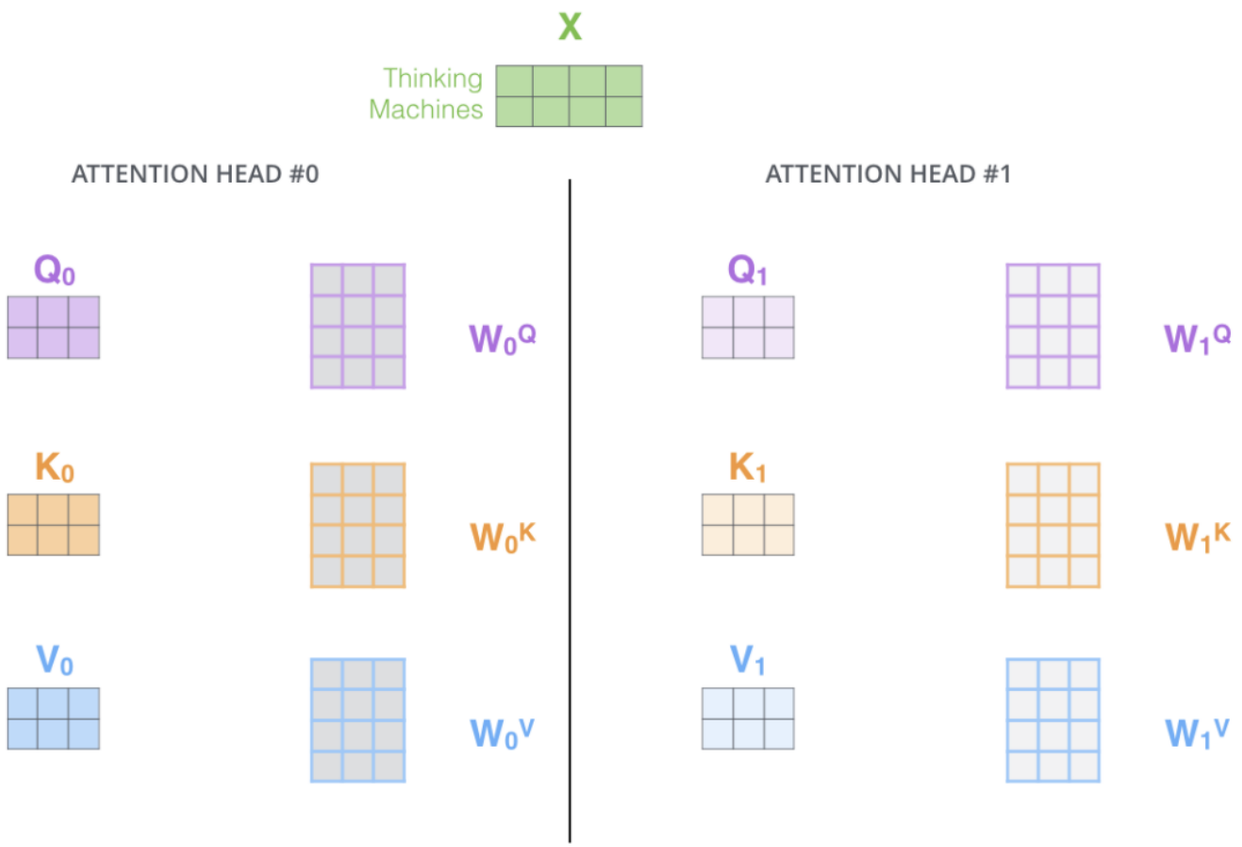
# 实际代码使用矩阵，方便并行



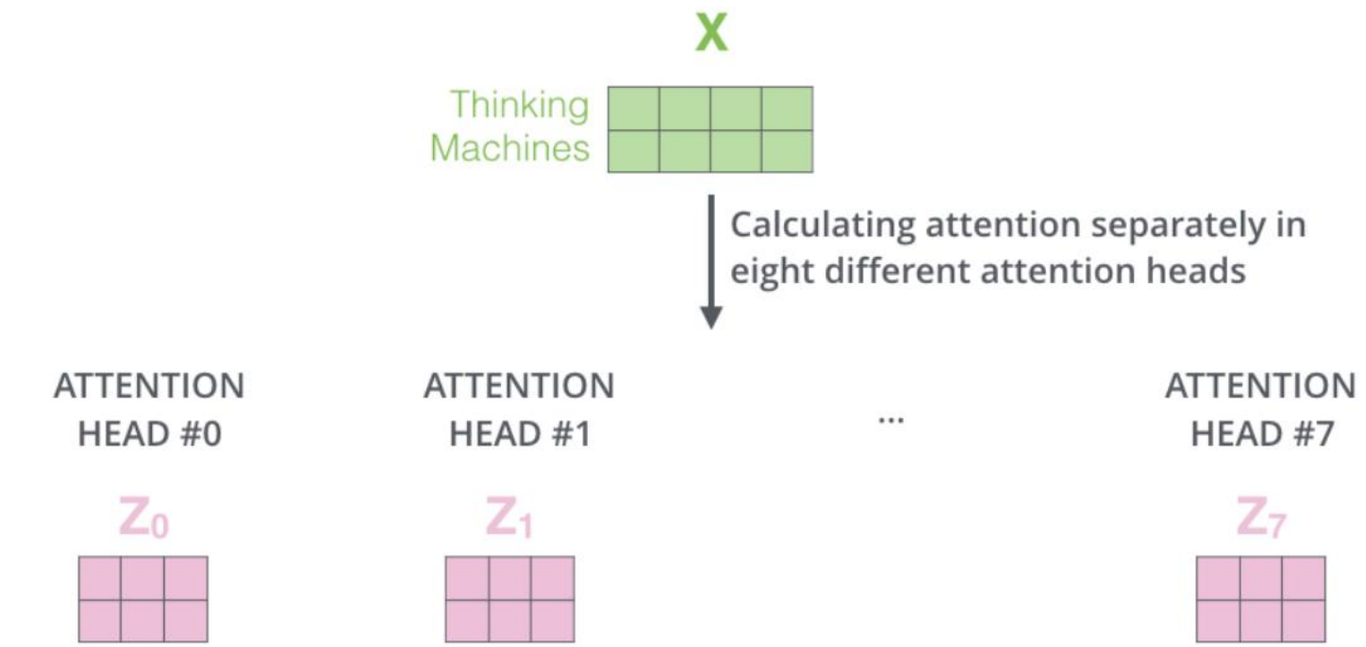




The self-attention calculation in matrix form

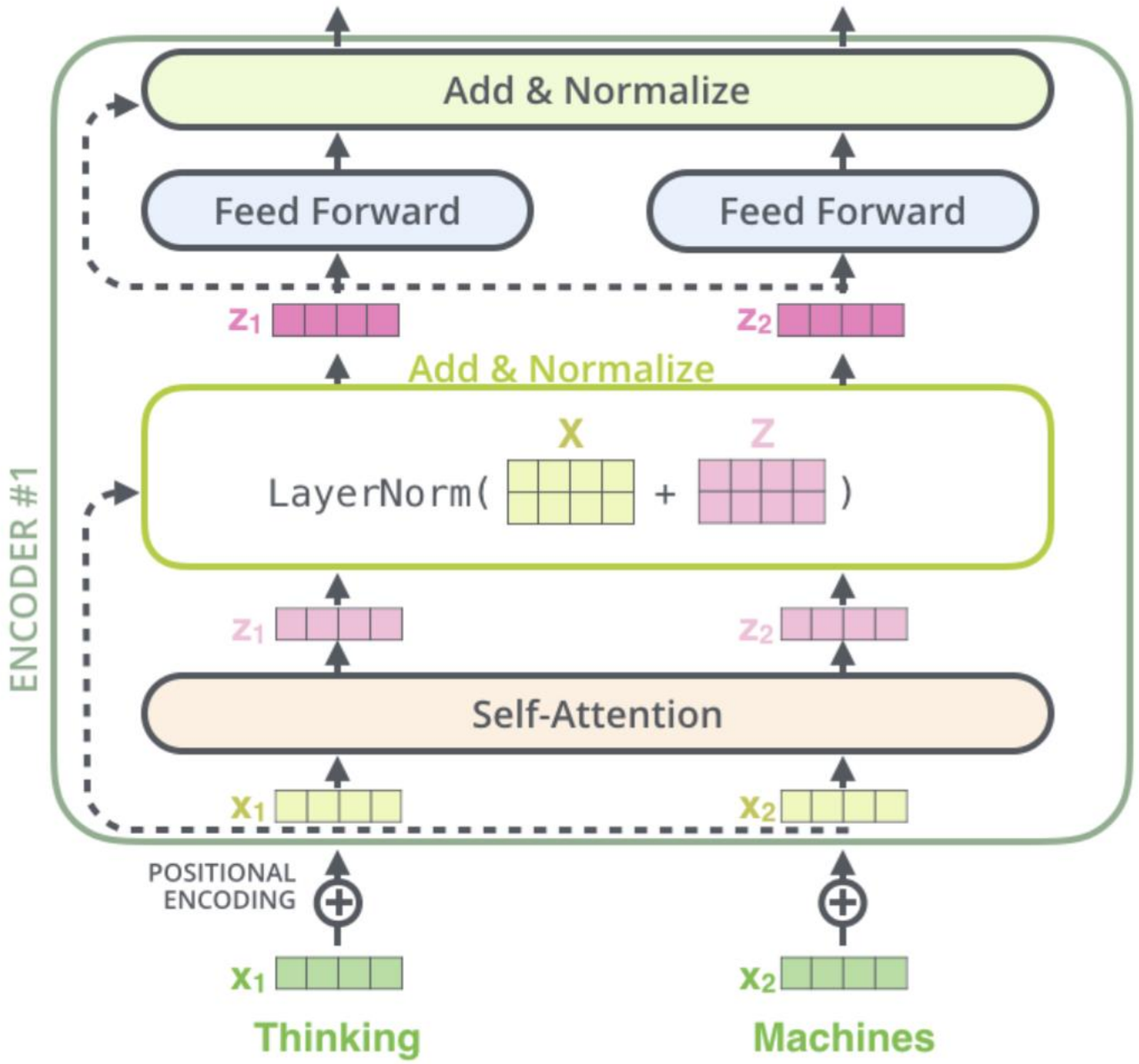
# 多头注意力机制


X
Thinking
Machines
ATTENTION HEAD #0
ATTENTION HEAD #1
Q0
W0Q
Q1
W1Q
K0
W0K
K1
W1K
V0
W0V
V1
W1V

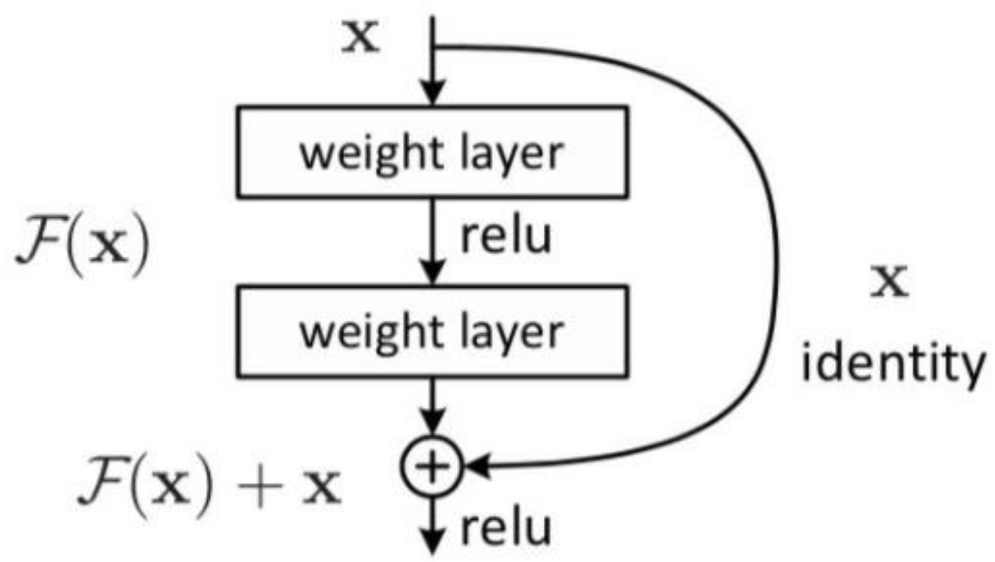
多个头就会有多个输出，需要合在一起输出



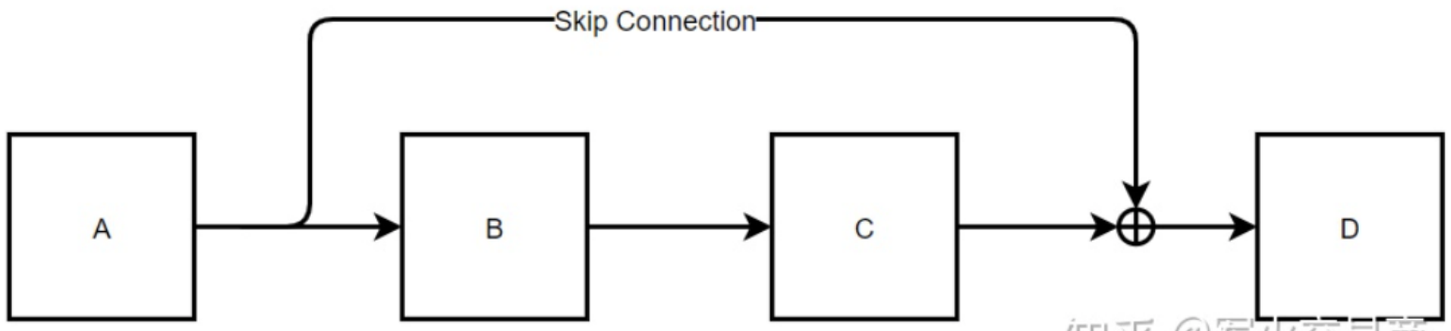
# 残差和LayNorm


Add & Normalize
Feed Forward
Feed Forward
z1
z2
Add & Normalize
X
Z
LayerNorm( + )
z1
z2
Self-Attention
x1
x2
ENCODER #1
POSITIONAL ENCODING
x1
x2
Thinking
Machines

# 残差

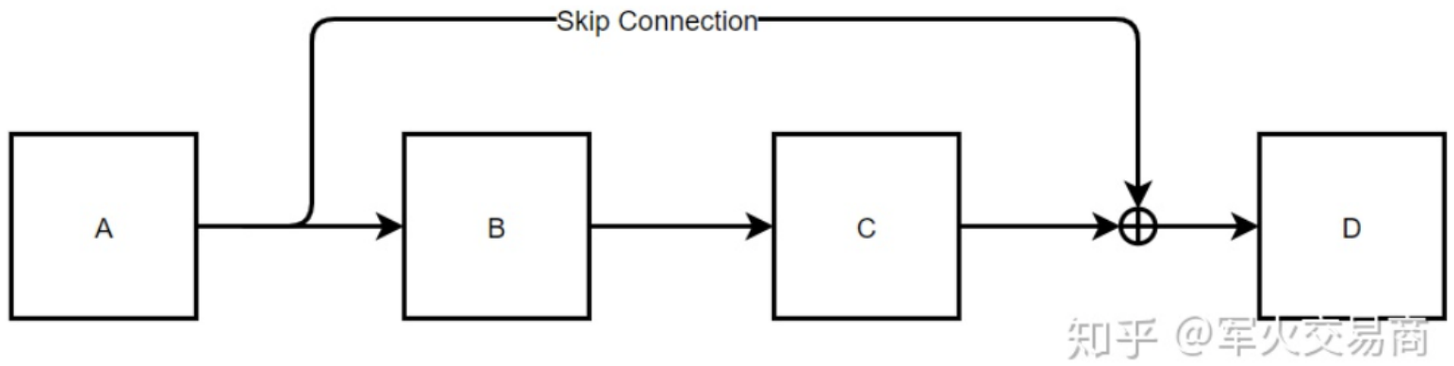

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

# 残差的作用






A,B,C,D 为四个不同的网络块，箭头代表"数据流"




A,B,C,D 为四个不同的网络块，箭头代表"数据流"

根据后向传播的链式法则，

$$\frac{\partial L}{\partial X_{Aout}} = \frac{\partial L}{\partial X_{Din}} \frac{\partial X_{Din}}{\partial X_{Aout}}$$

而 $X_{Din} = X_{Aout} + C(B(X_{Aout}))$

所以:

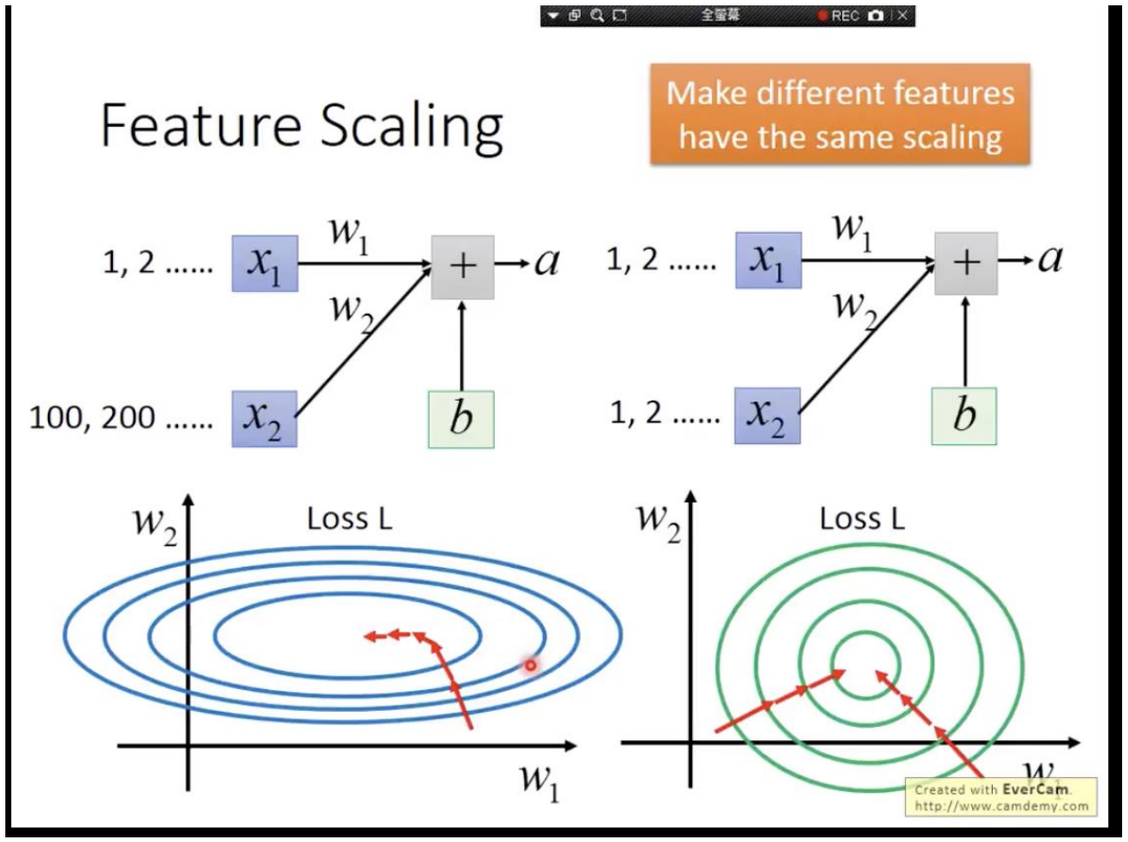
$$\frac{\partial L}{\partial X_{Aout}} = \frac{\partial L}{\partial X_{Din}} [1 + \frac{\partial X_{Din}}{\partial X_C} \frac{\partial X_C}{\partial X_B} \frac{\partial X_B}{\partial X_{Aout}}]$$

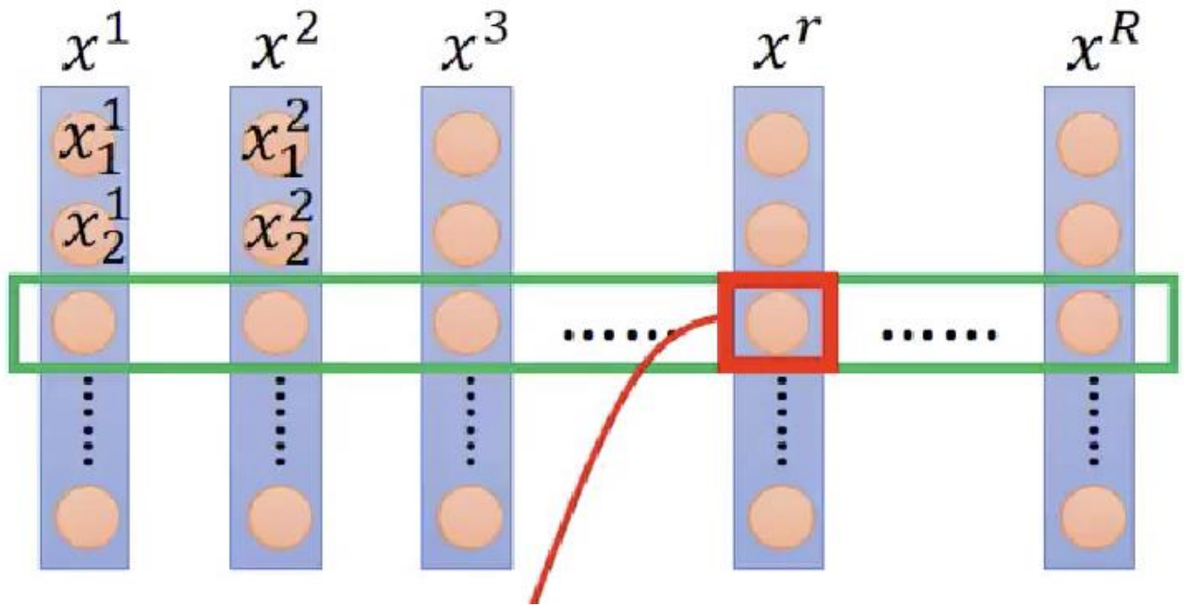
# 想一下RNN

# Layer Normalization

BN的效果差，所以不用

# 什么是BN，以及使用场景




$x^1$
$x^2$
$x^3$
$x^r$
$x^R$
$x_1^1$
$x_2^1$
$x_1^2$
$x_2^2$

# BN优点

第一个就是可以解决内部协变量偏移

第二个优点就是缓解了梯度饱和问题（如果使用sigmoid激活函数的话），加快收敛

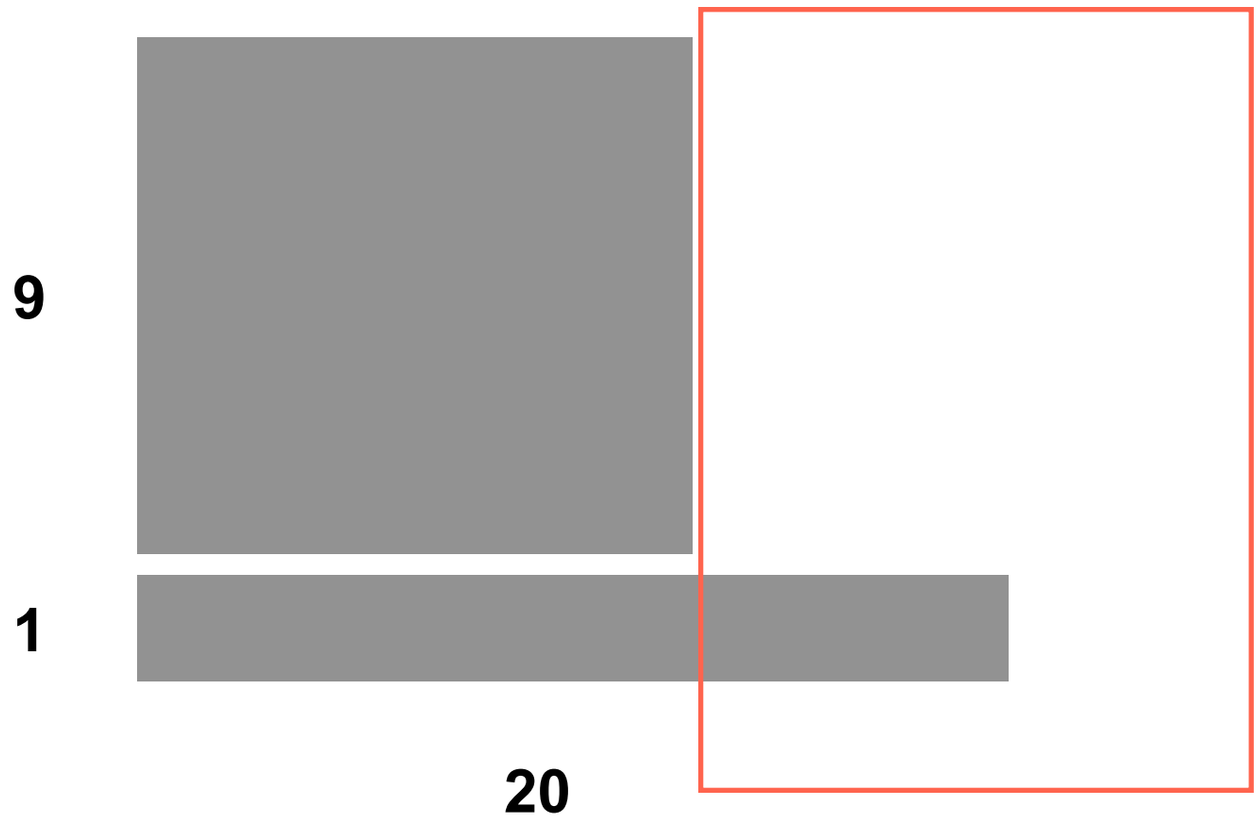
## BN的缺点

第一个，batch_size较小的时候，效果差。

## BN的缺点

第二个缺点就是 BN 在RNN中效果比较差。这一点和第一点原因很类似，不过我单挑出来说。

5



为什么使用layer-norm

**理解**：为什么LayerNorm单独对一个样本的所有单词做缩放可以起到效果。

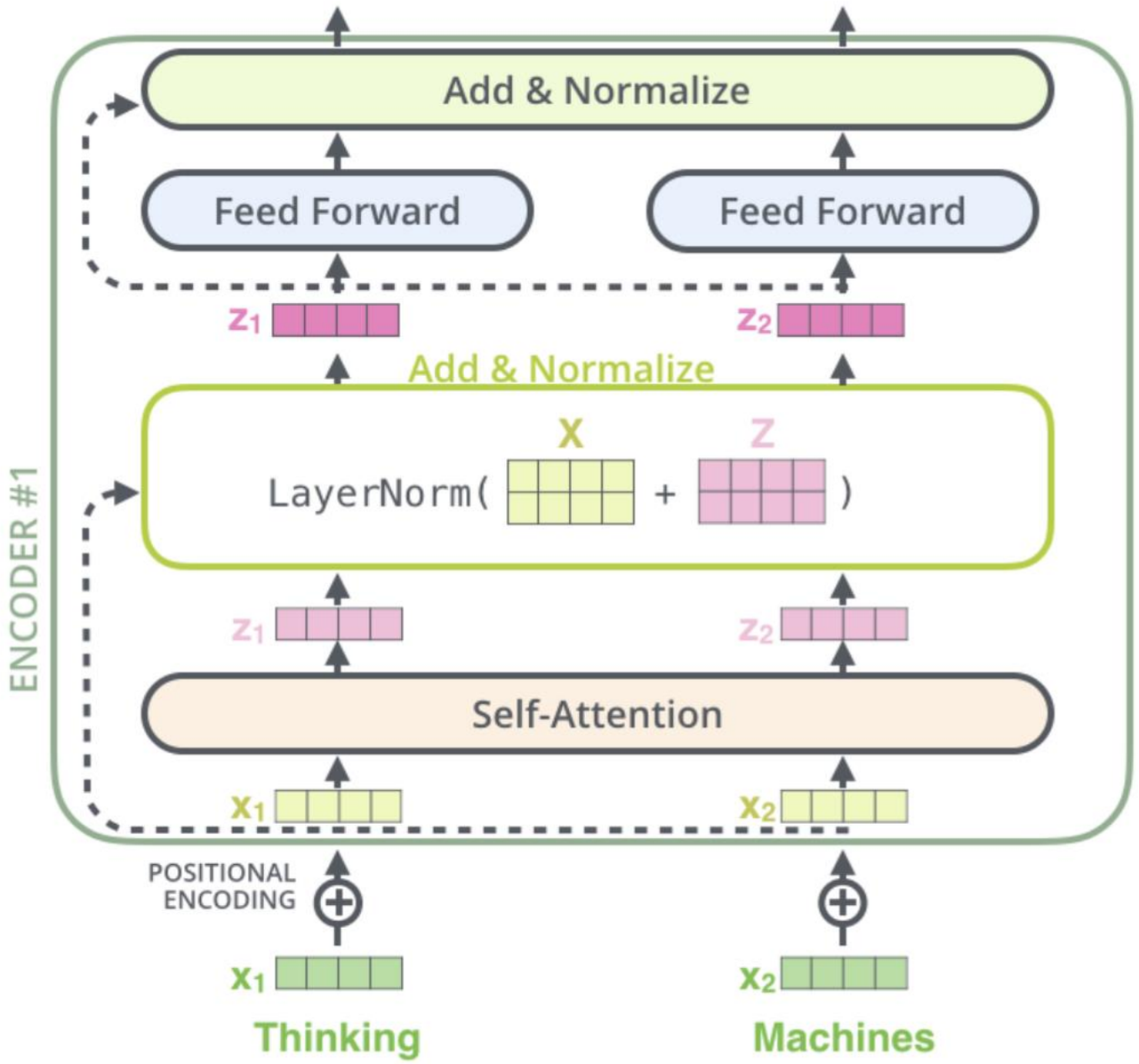
# 把BN引申到RNN

我爱中国共产党

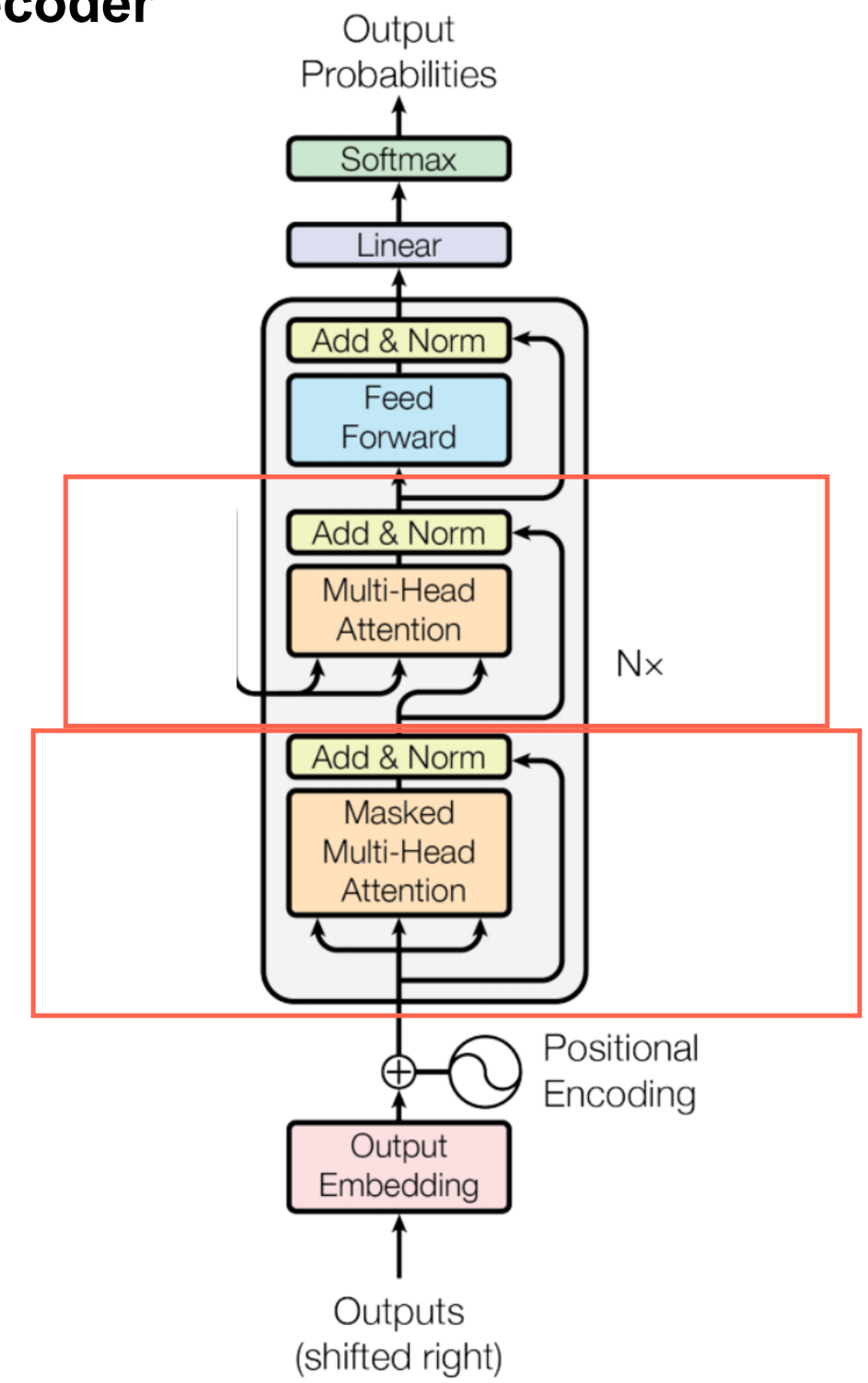


今天天气真不错

## 前馈神经网路


Add & Normalize
Feed Forward
Feed Forward
z1
z2
Add & Normalize
X
Z
LayerNorm( + )
z1
z2
Self-Attention
x1
x2
ENCODER #1
POSITIONAL ENCODING
x1
x2
Thinking
Machines

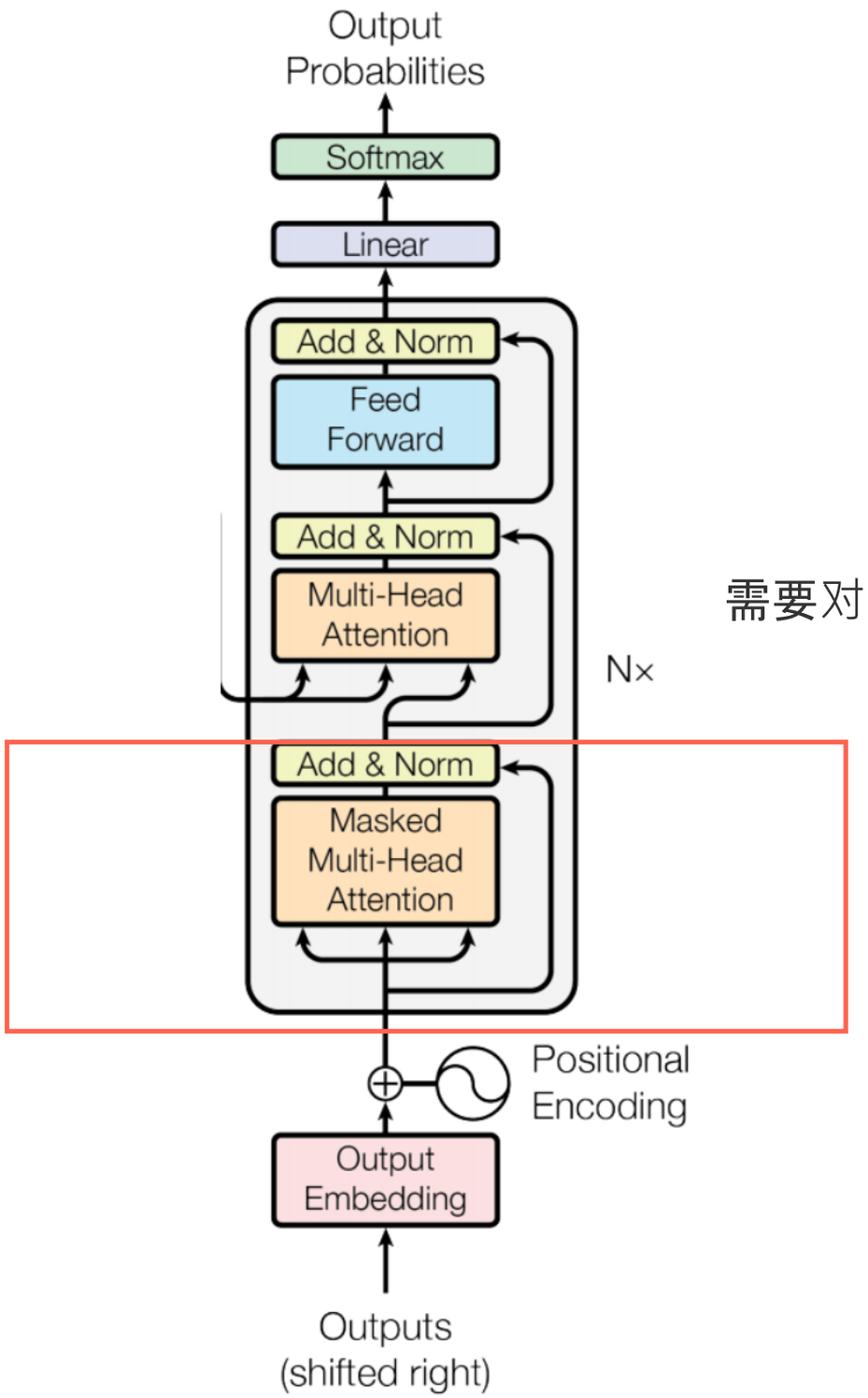
# Decoder


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

2

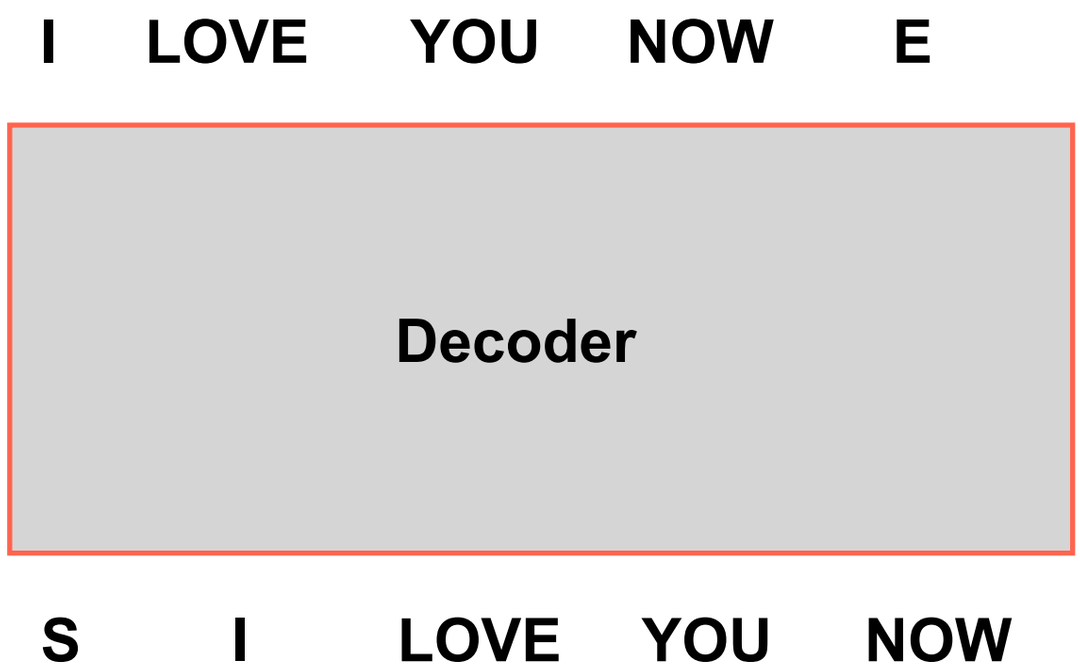
1

## 1：多头注意力机制

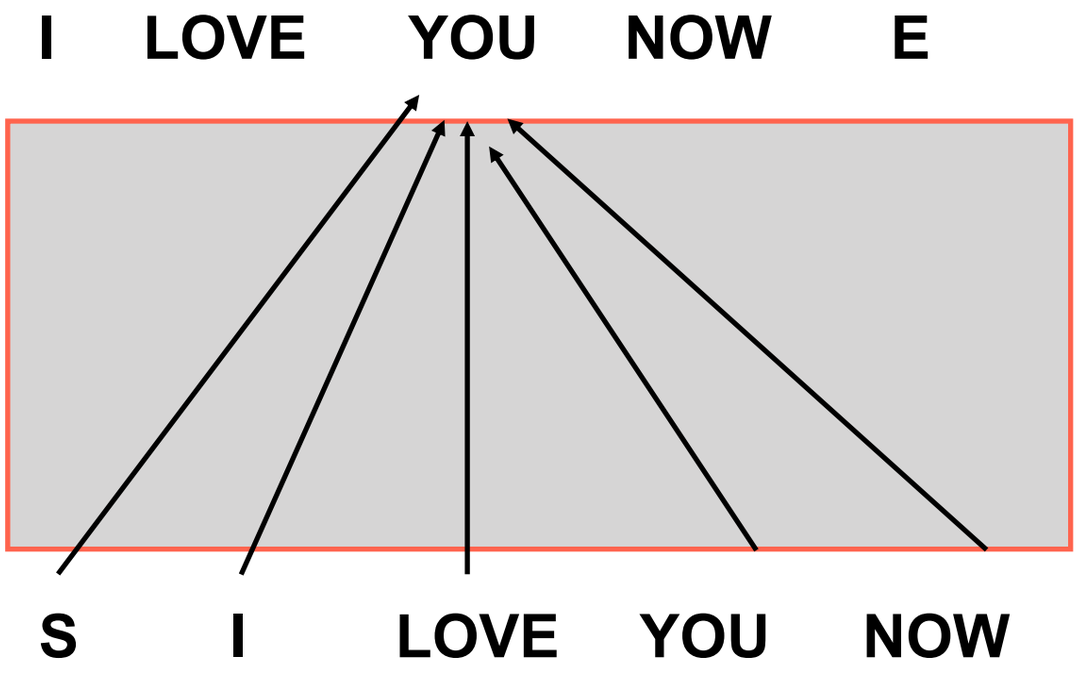



需要对当前单词和之后的单词做mask。

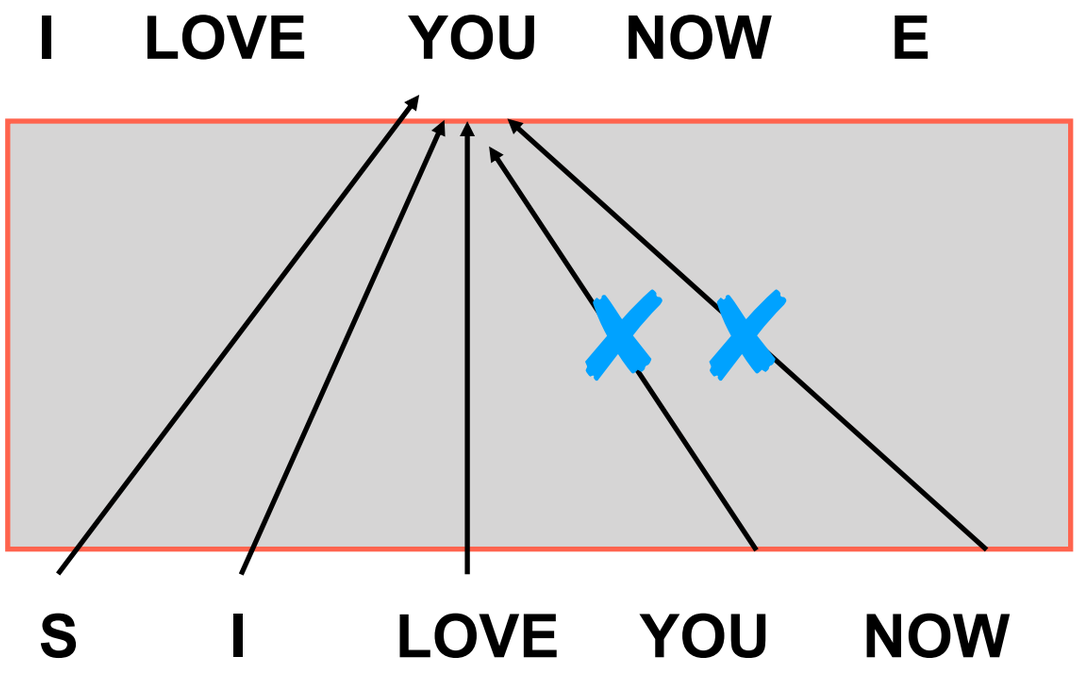
为什么需要mask



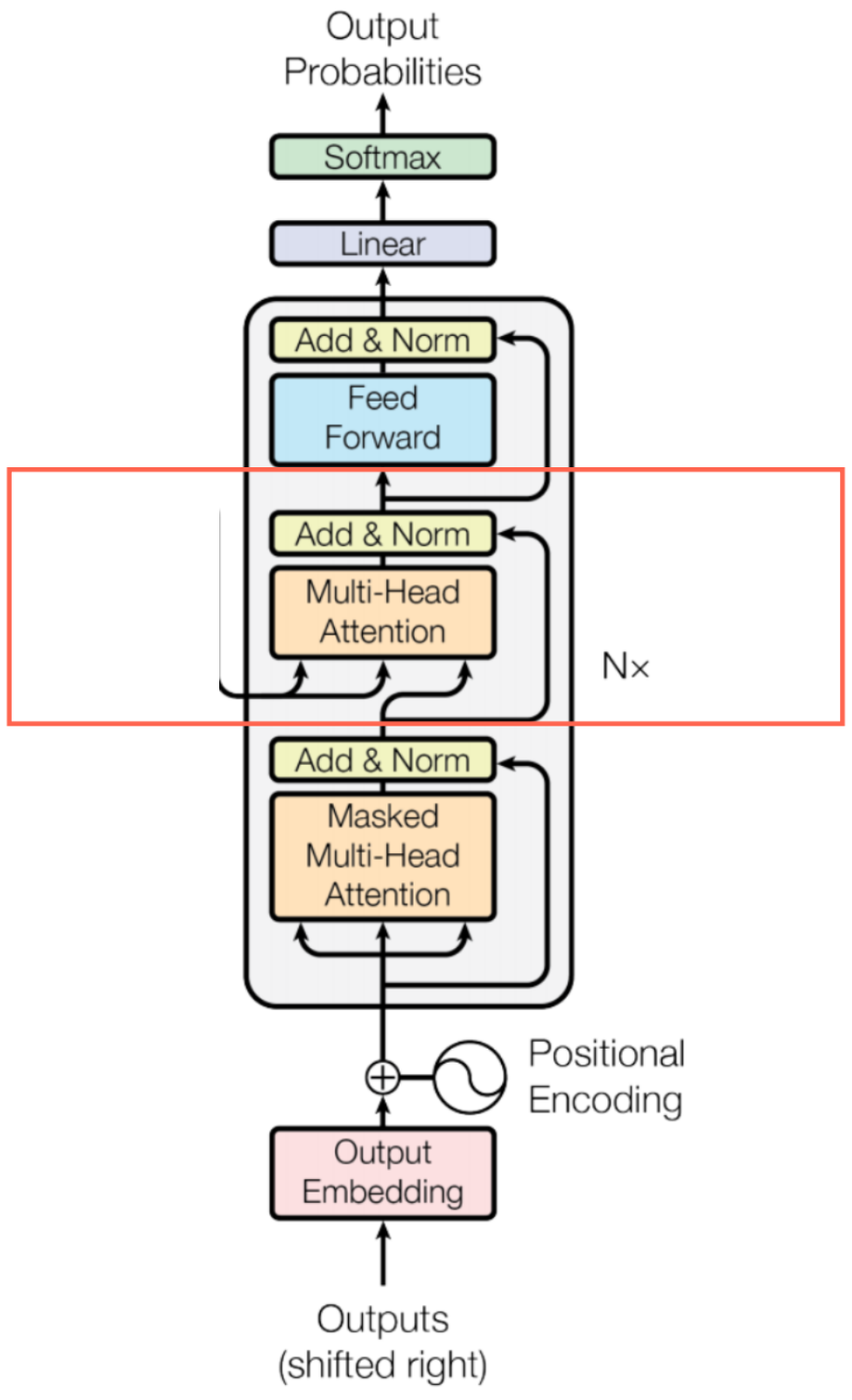
为什么需要mask

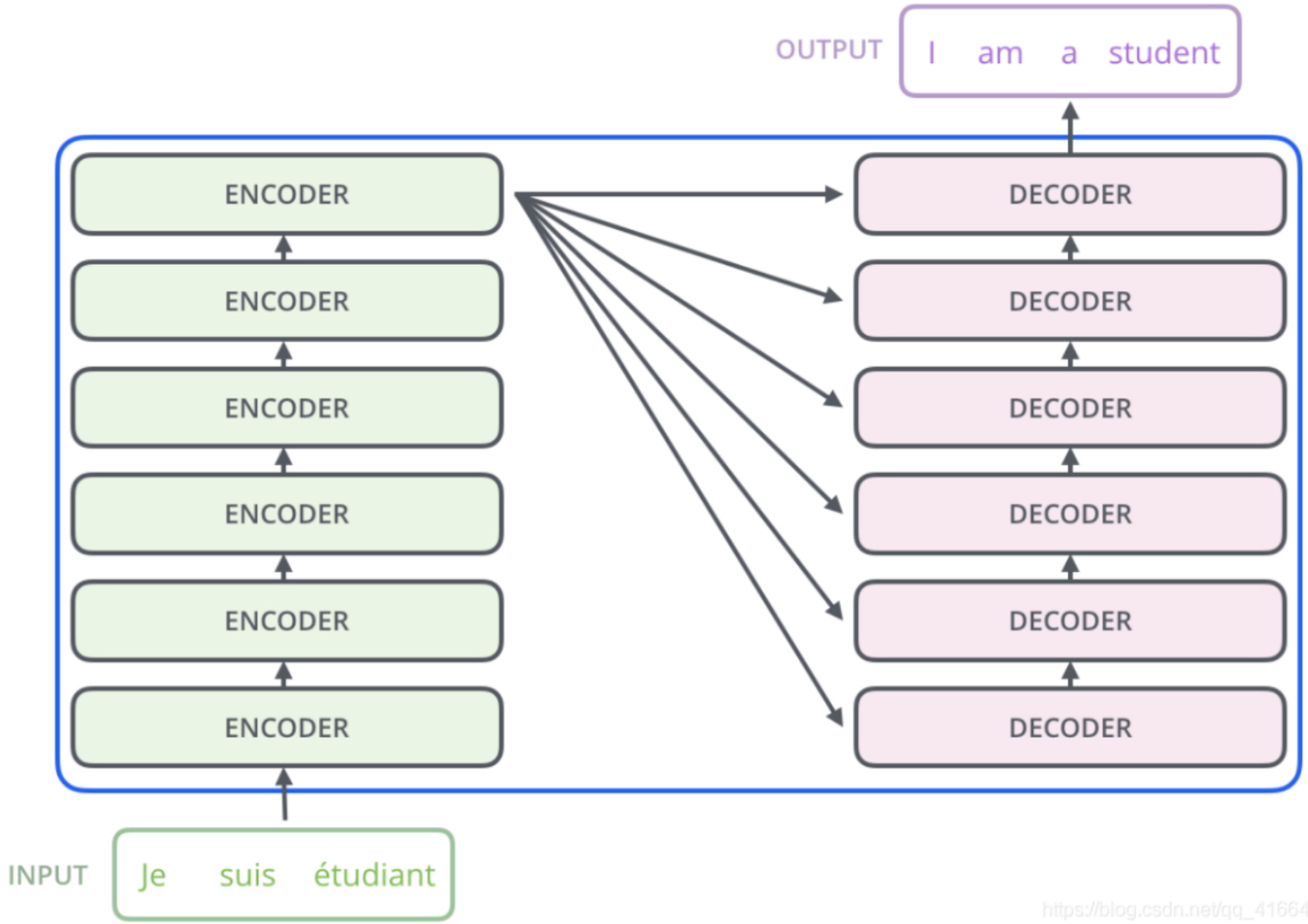


为什么需要mask

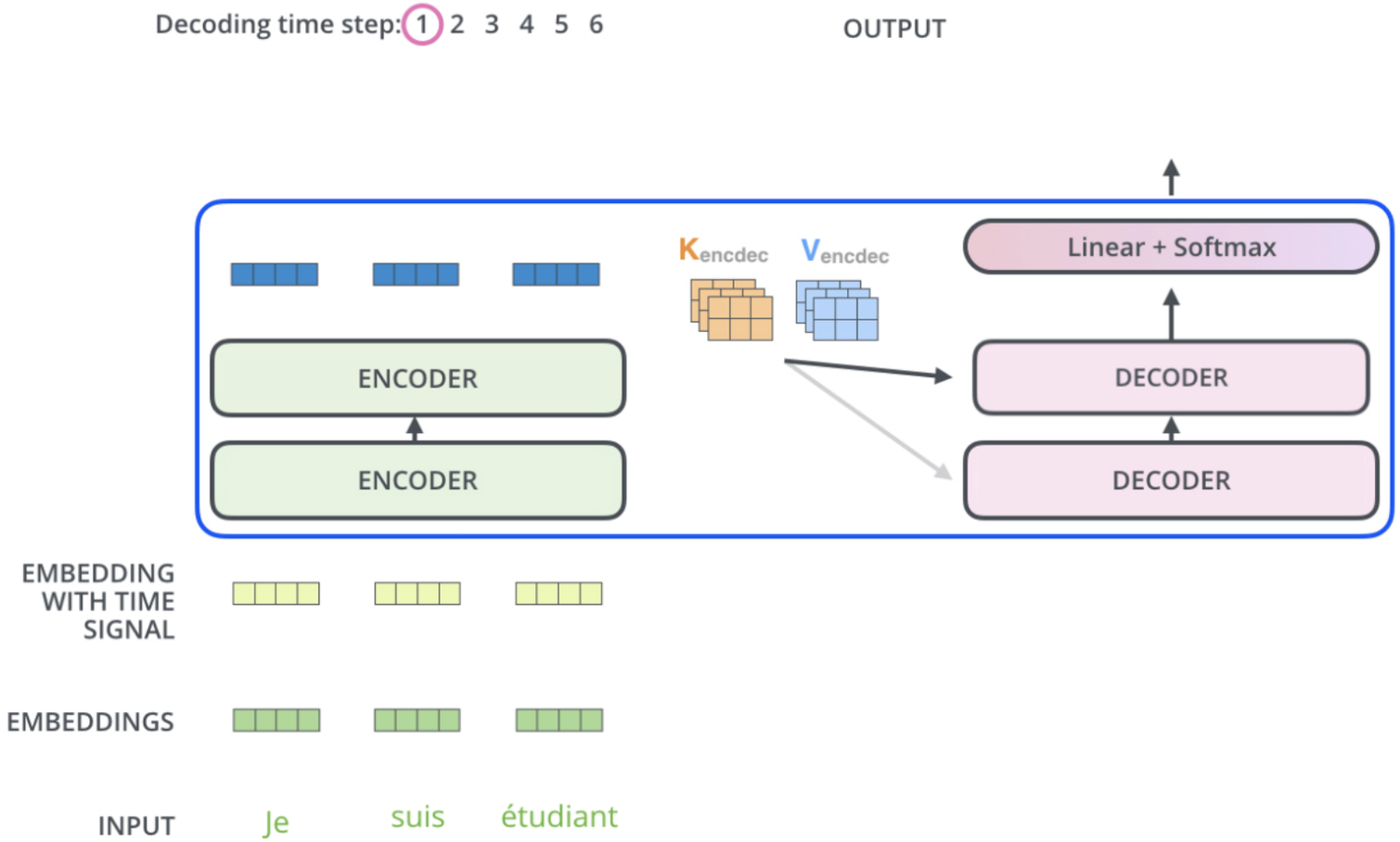


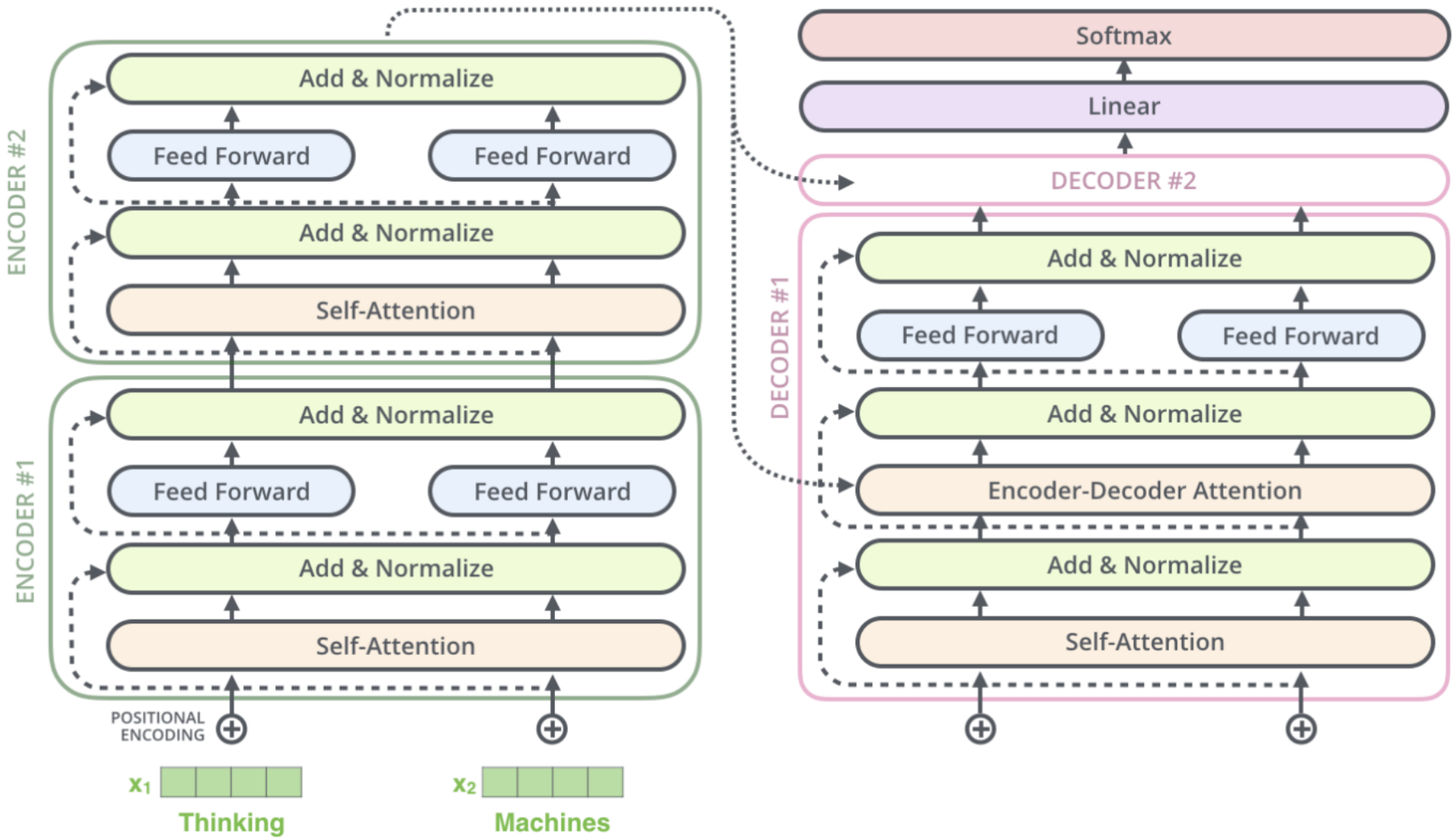
# 交互层


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



# Decoder


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


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

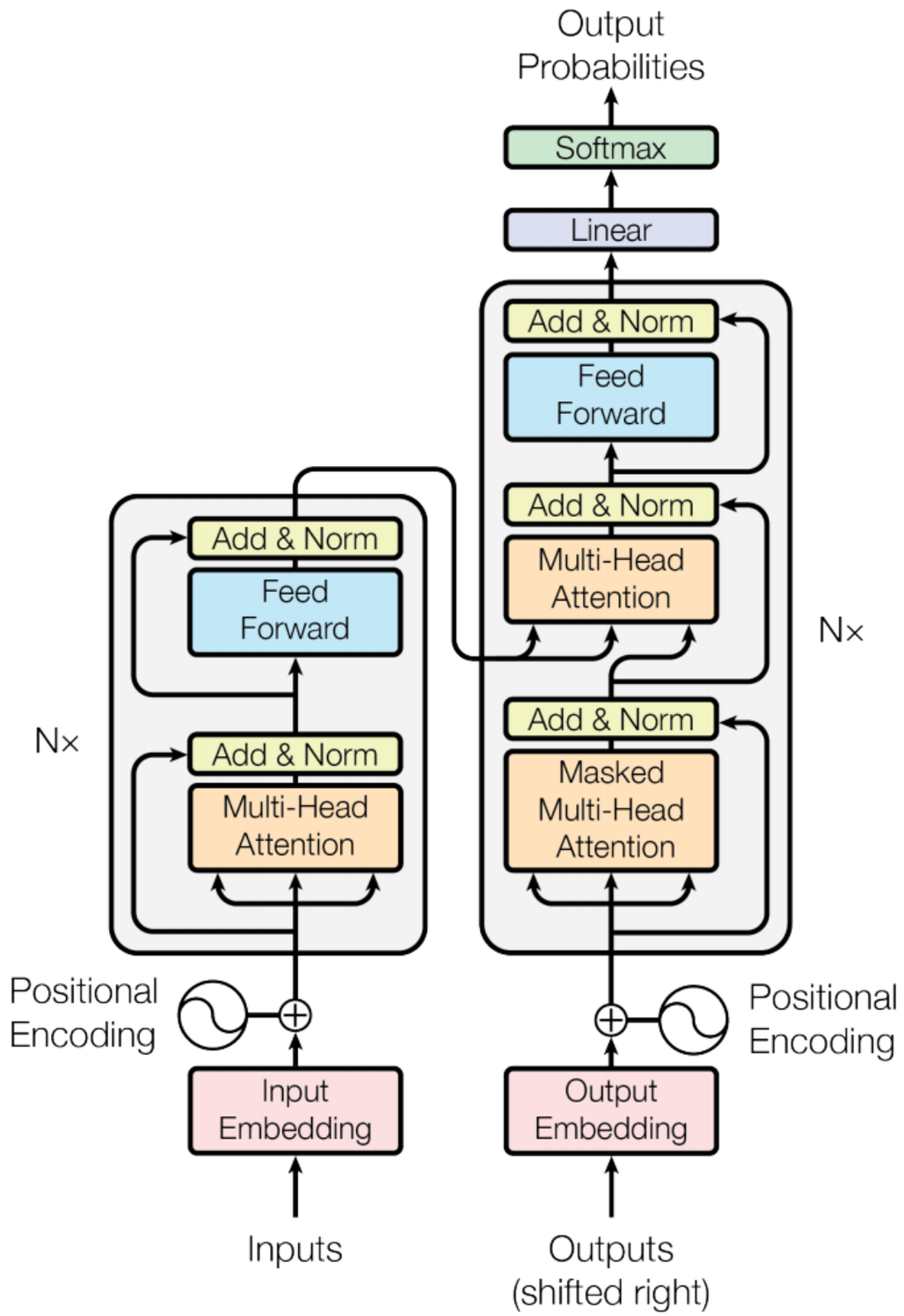
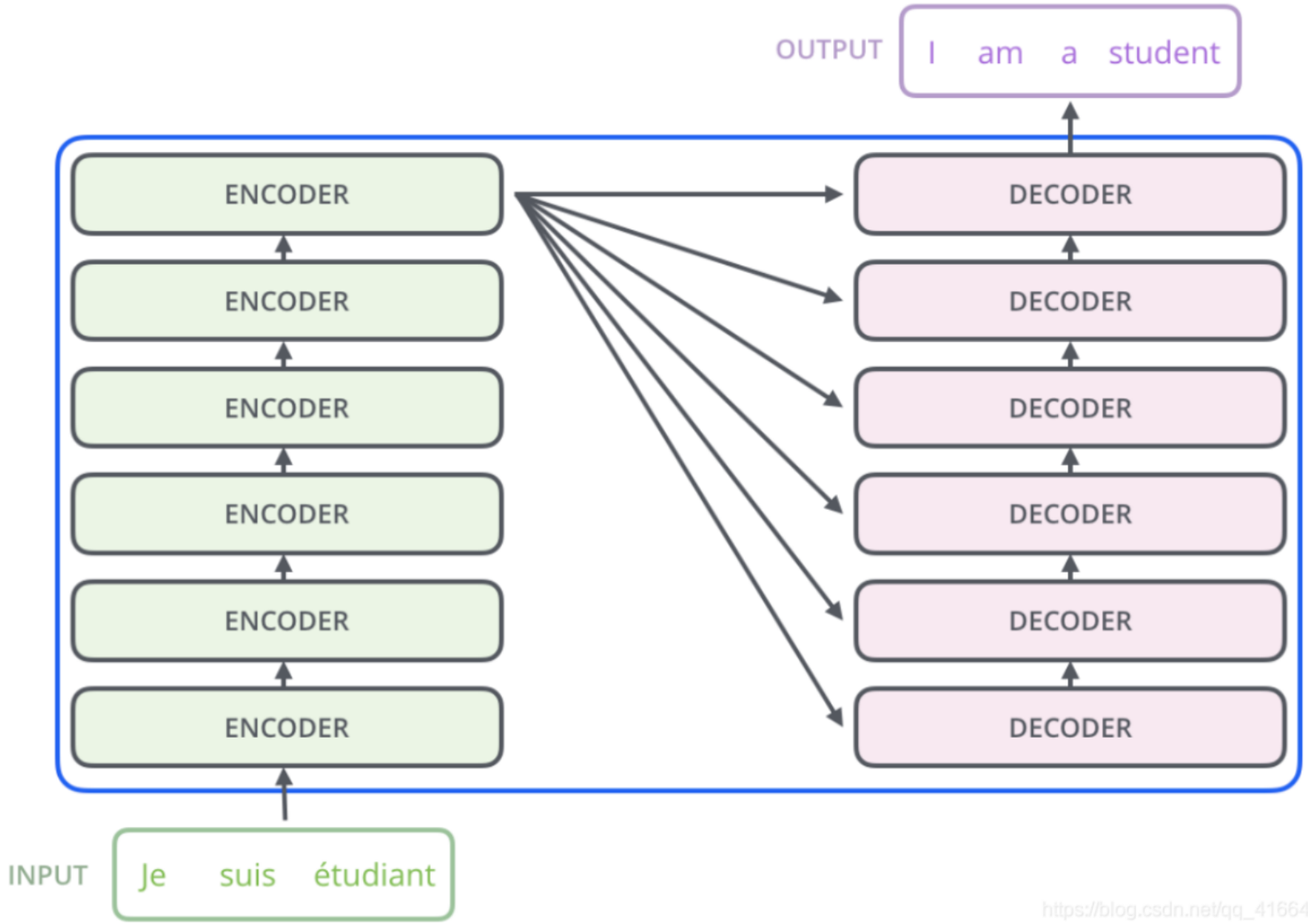
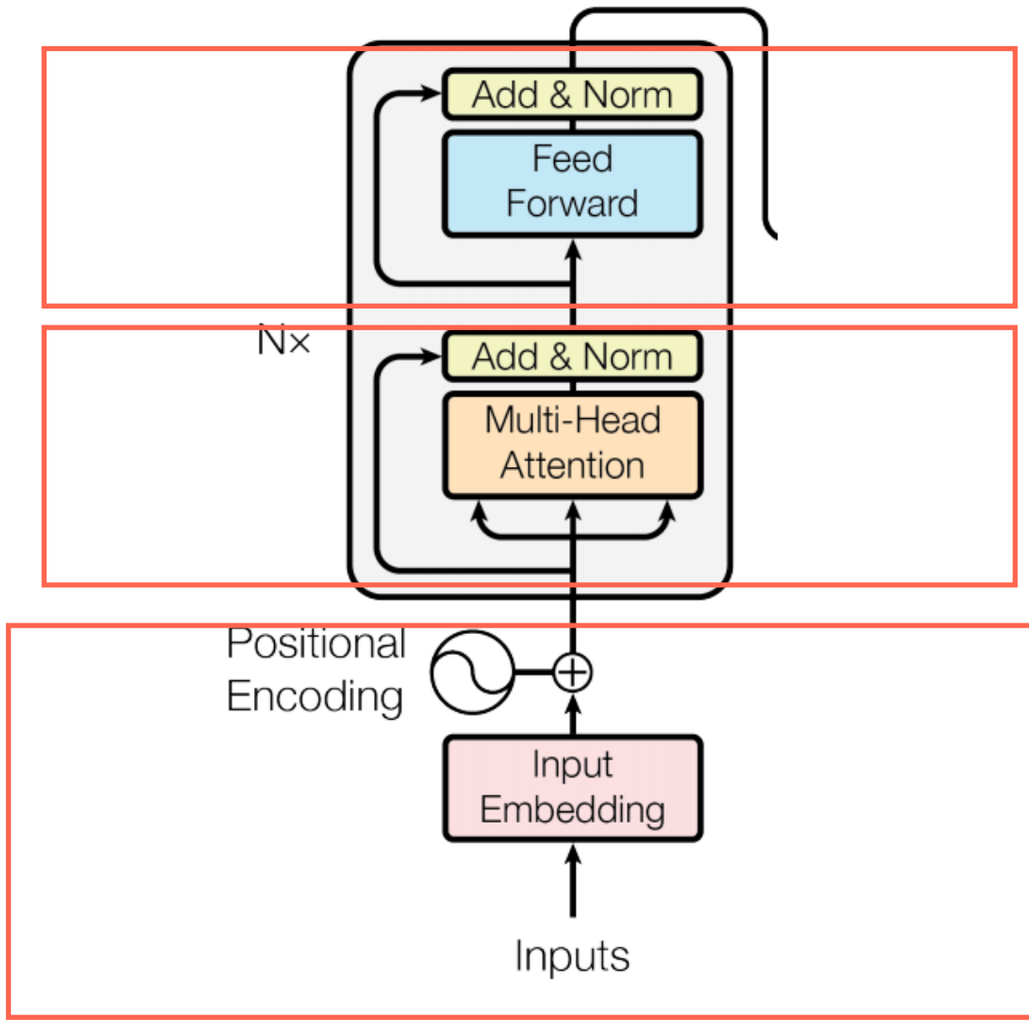



Figure 1: The Transformer - model architecture.






3 前馈神经网络

2 注意力机制

1 输入部分

# 元