

Attention is All you need

目录

- 背景动机
- 模型架构
- 模型细节
- 实验结果

背景动机

Rnn和Lstm存在下面两个问题

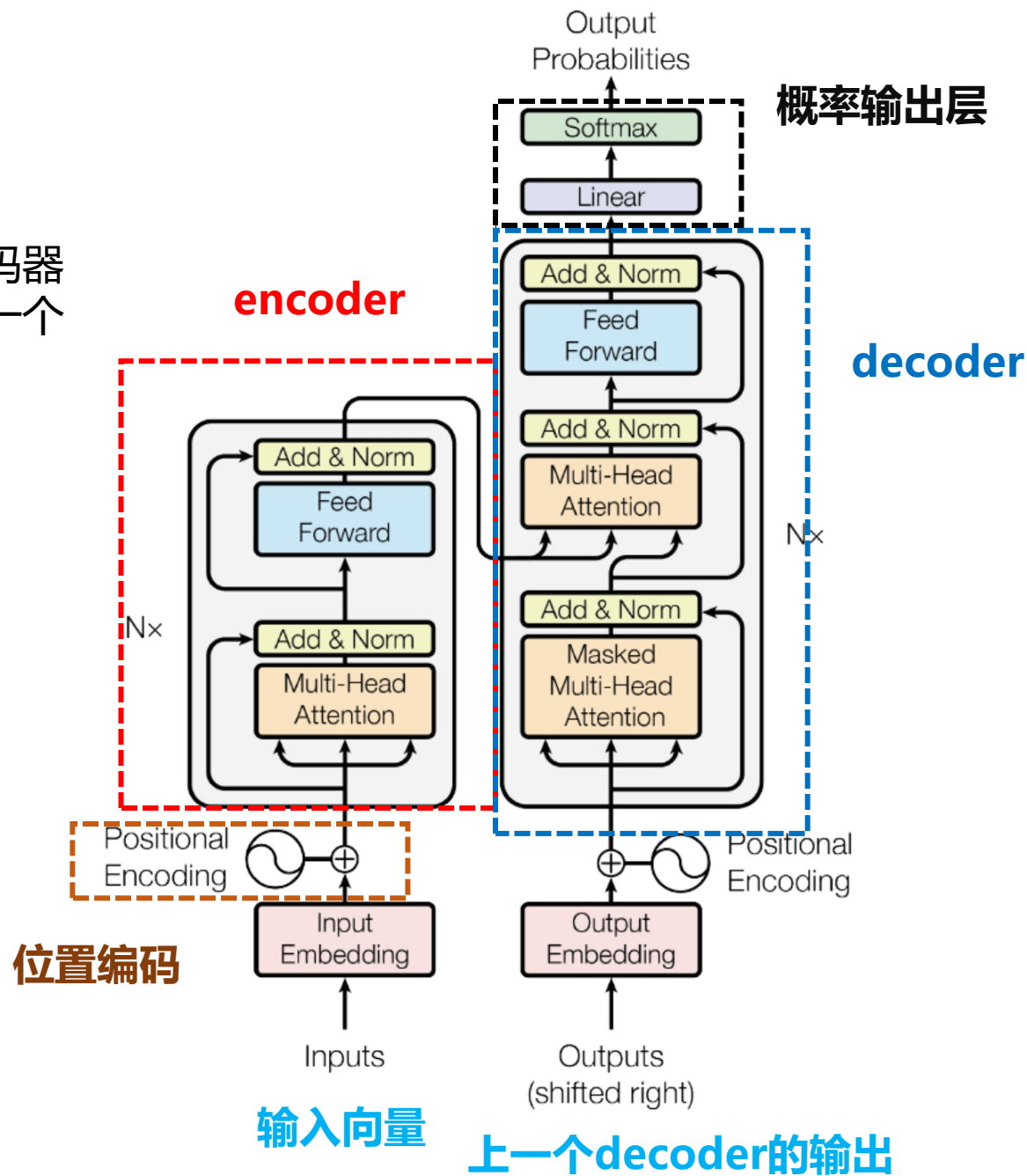
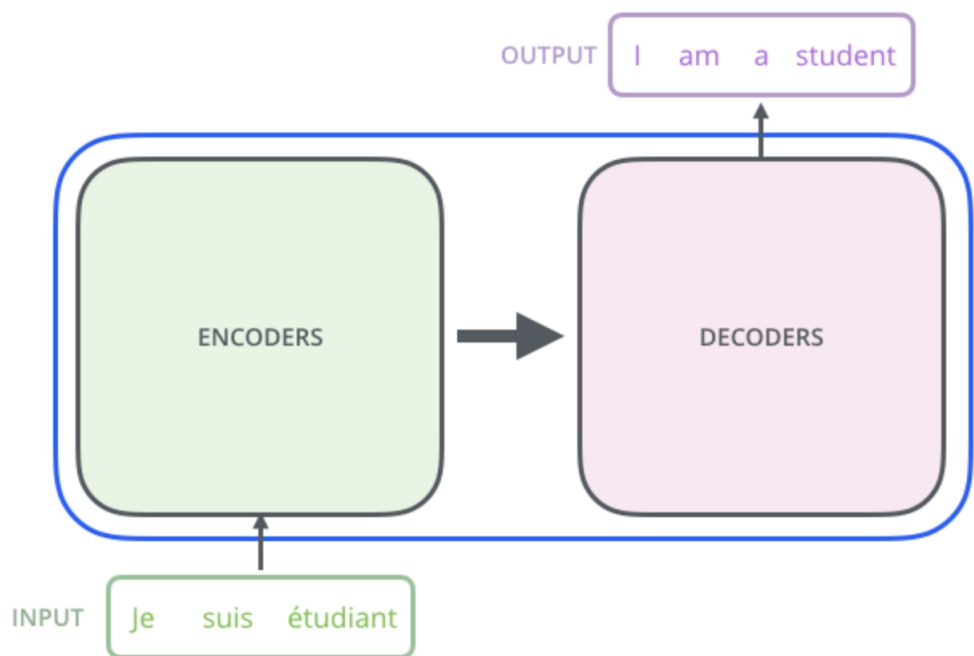
- 1.时间 t 的计算依赖 $t-1$ 时刻的计算结果，这样限制了模型的并行能力；
- 2.顺序计算的过程中信息会丢失，尽管LSTM等门机制的结构一定程度上缓解了长期依赖的问题，
- 3.但是对于特别长期的依赖现象,LSTM依旧无能为力。

解决方案

使用Attention机制，将序列中的任意两个位置之间的距离是缩小为一个常量；
具有更好的并行性，符合现有的GPU框架。

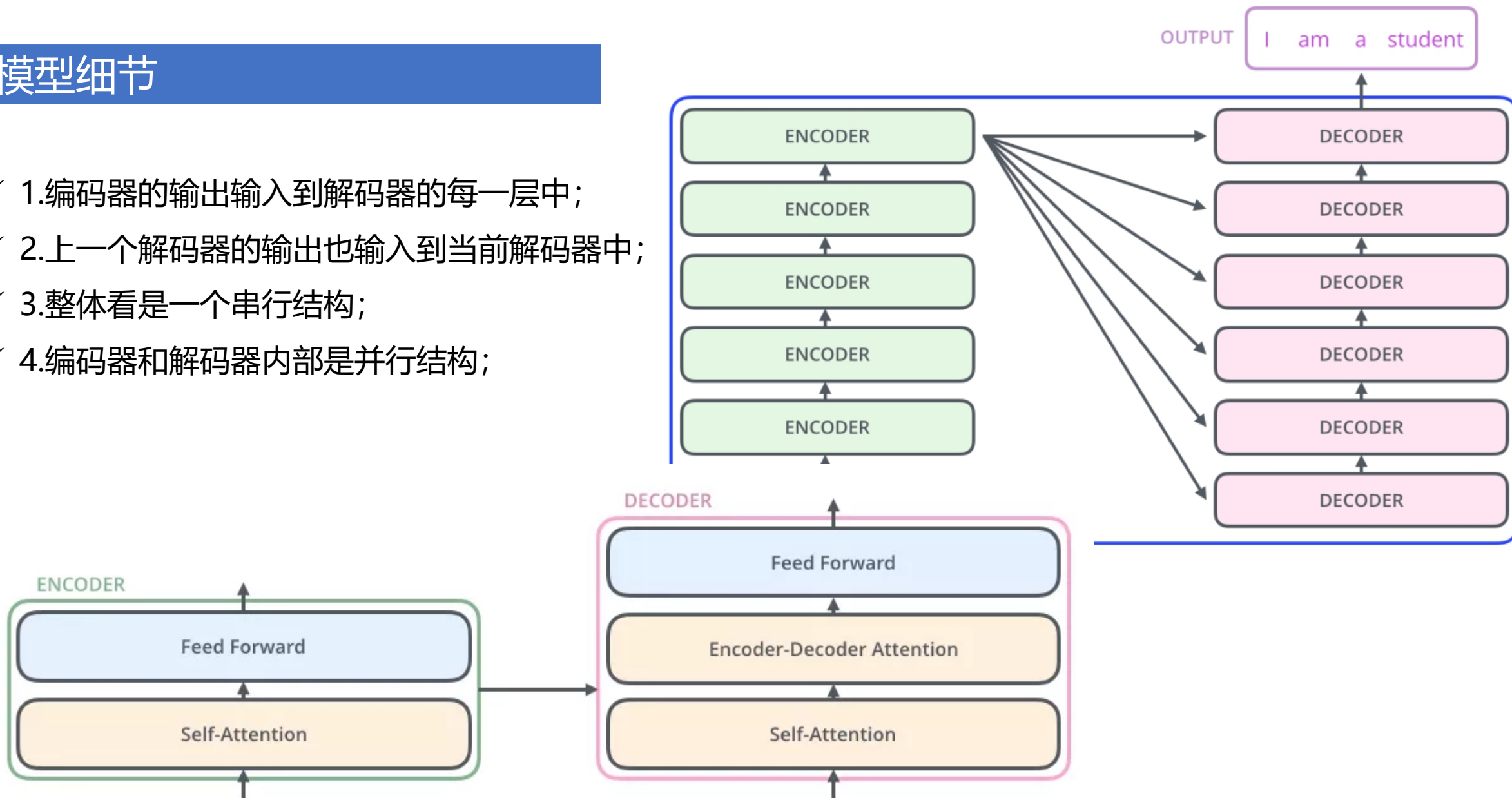
模型架构

主要结构有：编码器Encoder 和 解码器Decoder
输入向量融合位置编码函数，之后经过N个编码器编码，
然后输出两个向量到解码器中间的多头注意层中。而解码器的输入有两部分，一部分是编码器的输出，一部分是上一个解码器的输出融合位置编码函数的输出。



模型细节

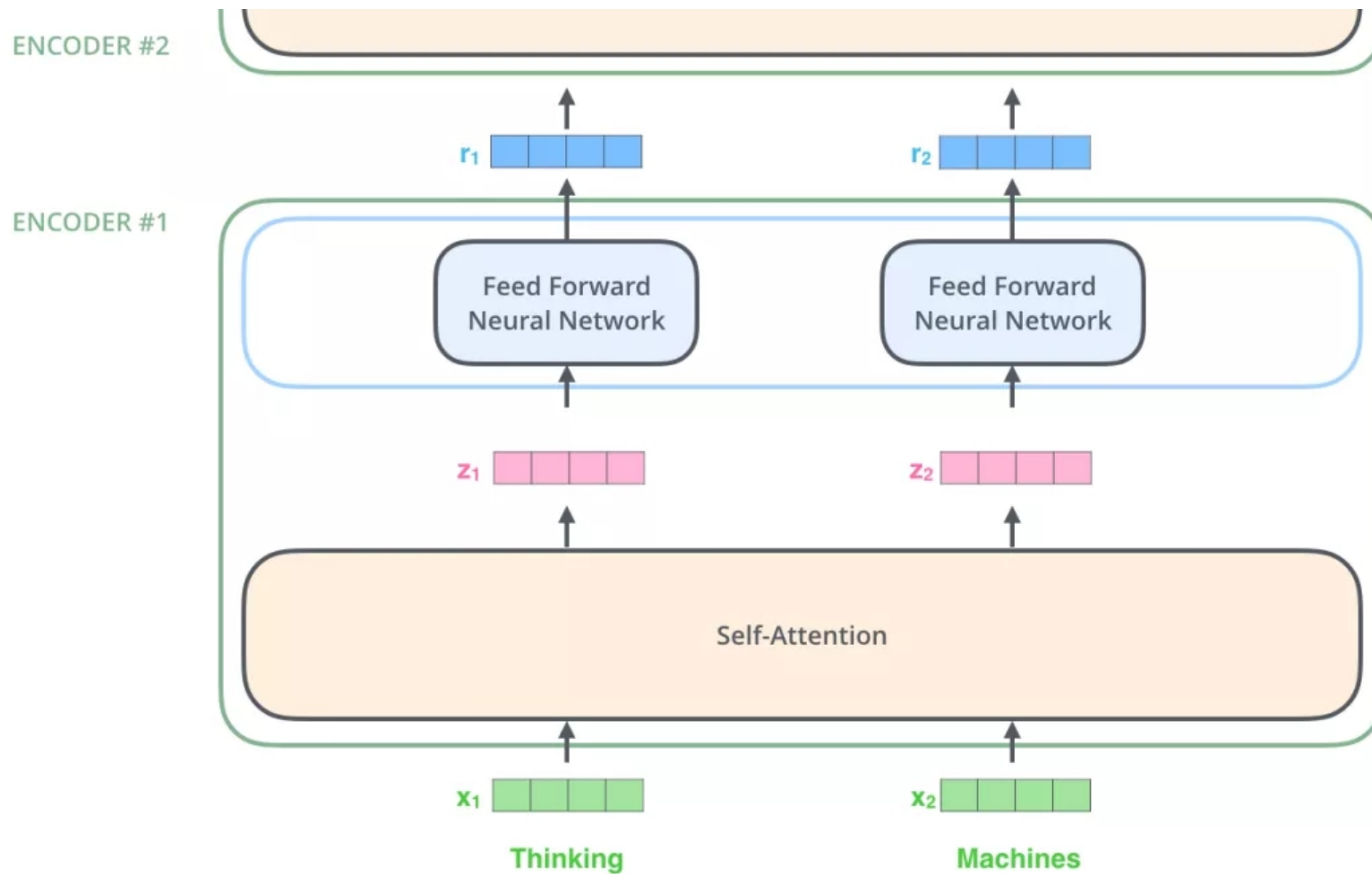
- ✓ 1. 编码器的输出输入到解码器的每一层中;
- ✓ 2. 上一个解码器的输出也输入到当前解码器中;
- ✓ 3. 整体看是一个串行结构;
- ✓ 4. 编码器和解码器内部是并行结构;



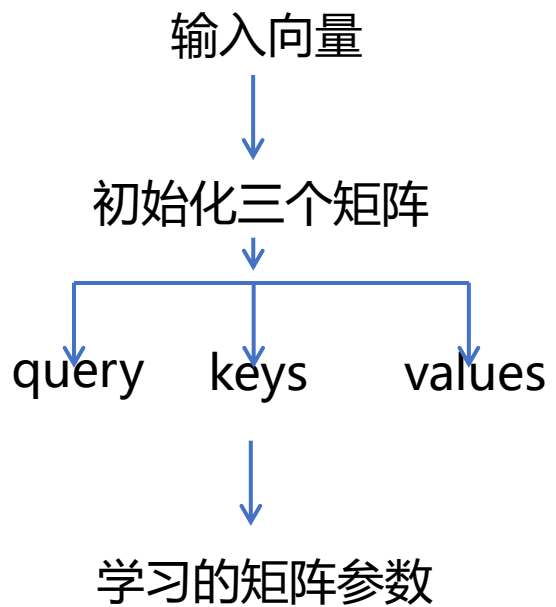
模型细节

**X1,x2经过self attention和
feed forward都是
并行计算的**

整个输入的句子是一个向量列表
其中有 2 个词向量 x_1, x_2



模型架构



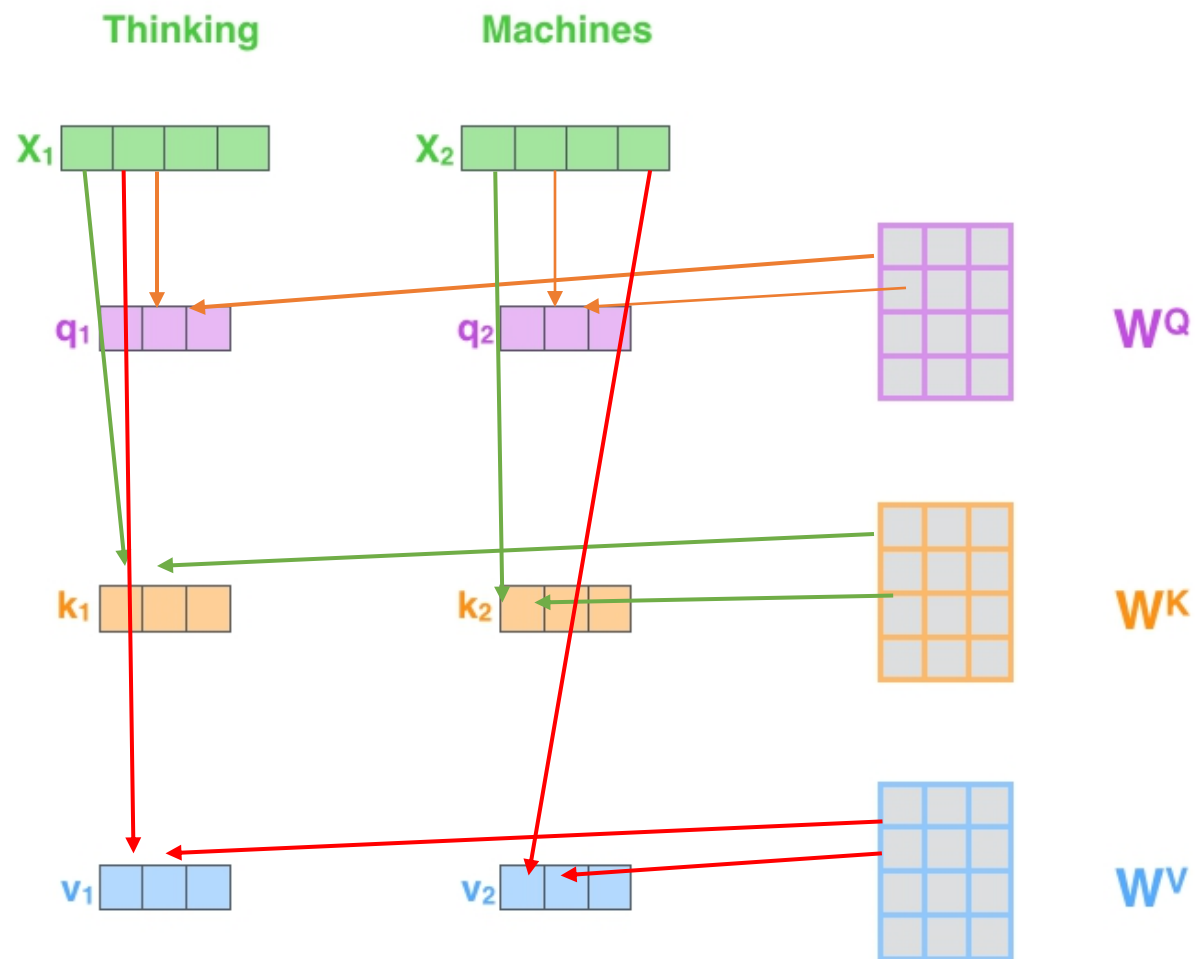
Input

Embedding

Queries

Keys

Values



模型细节

注意力分数：Attention Score

计算对句子中其他位置的每个词放置多少的注意力。 $q_1 \cdot k_1$ 、 $q_1 \cdot k_2$

每个分数除以 （是 Key 向量的长度）

Softmax可以将分数归一化

将每个分数分别与每个 Value 向量相乘

对于分数高的位置，相乘后的值就越大

Input

Embedding

Queries

Keys

Values

Score

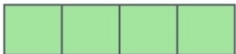
Divide by $8 (\sqrt{d_k})$

Softmax

Softmax
X
Value

Sum

Thinking

x_1 

q_1 

k_1 

v_1 

$q_1 \cdot k_1 = 112$

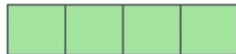
14

0.88

v_1 

z_1 

Machines

x_2 

q_2 

k_2 

v_2 

$q_1 \cdot k_2 = 96$

12

0.12

v_2 

z_2 

模型细节

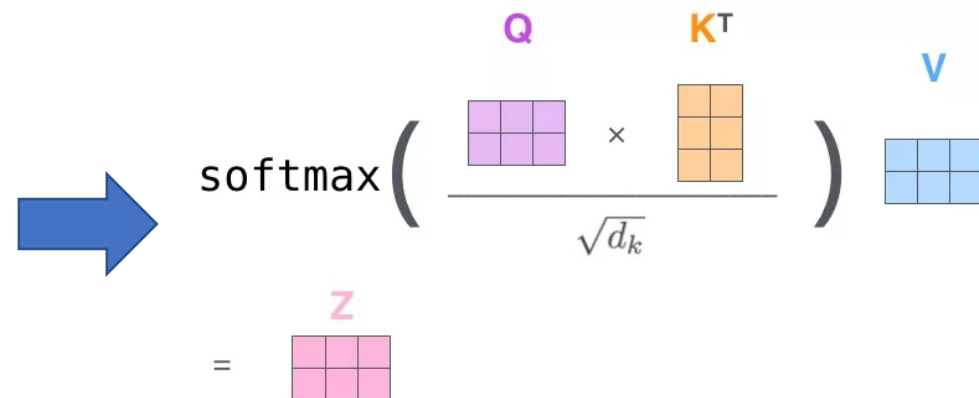
Query- 自己的信息



key- 与其他向量的
信息权重



Value- 区分注意力分数



多头注意力机制

目的：
一个self attention关注到的信息量不够，
需要多个一起协同合作

1. 把 8 个矩阵 $\{Z_0, Z_1, \dots, Z_7\}$ 拼接起来
2. 把拼接后的矩阵和 W^O 权重矩阵相乘
3. 得到最终的矩阵 Z ，这个矩阵包含了所有 attention heads（注意力头）的信息。这个矩阵会输入到 FFNN (Feed Forward Neural Network) 层。

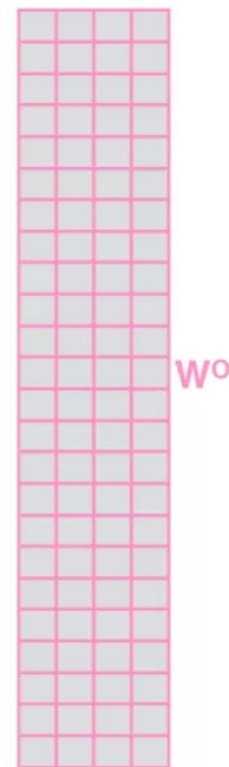
1) Concatenate all the attention heads



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

\times

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



整体情况

1) This is our input sentence*

2) We embed each word*

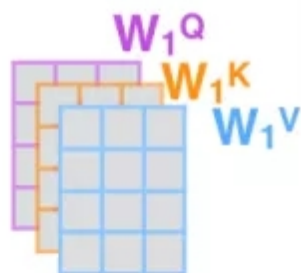
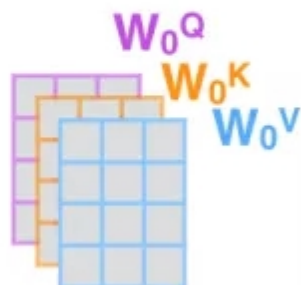
Thinking
Machines



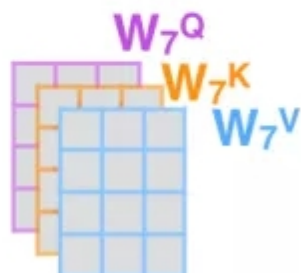
* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one



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



...



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



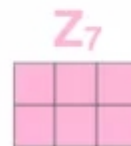
...



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



...



位置编码

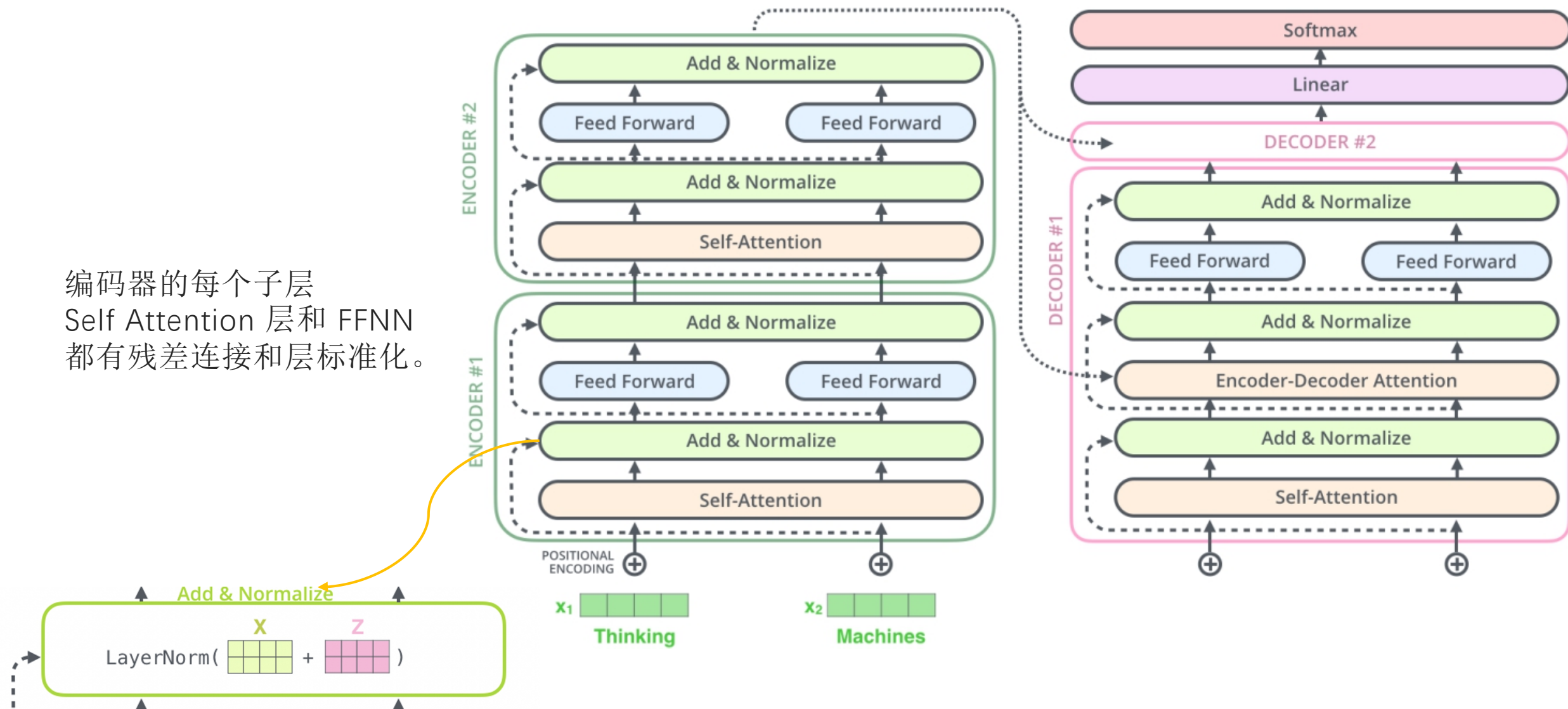
在输入向量上叠加位置编码。
目的是为了记住每个词向量在句子中的位置。

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



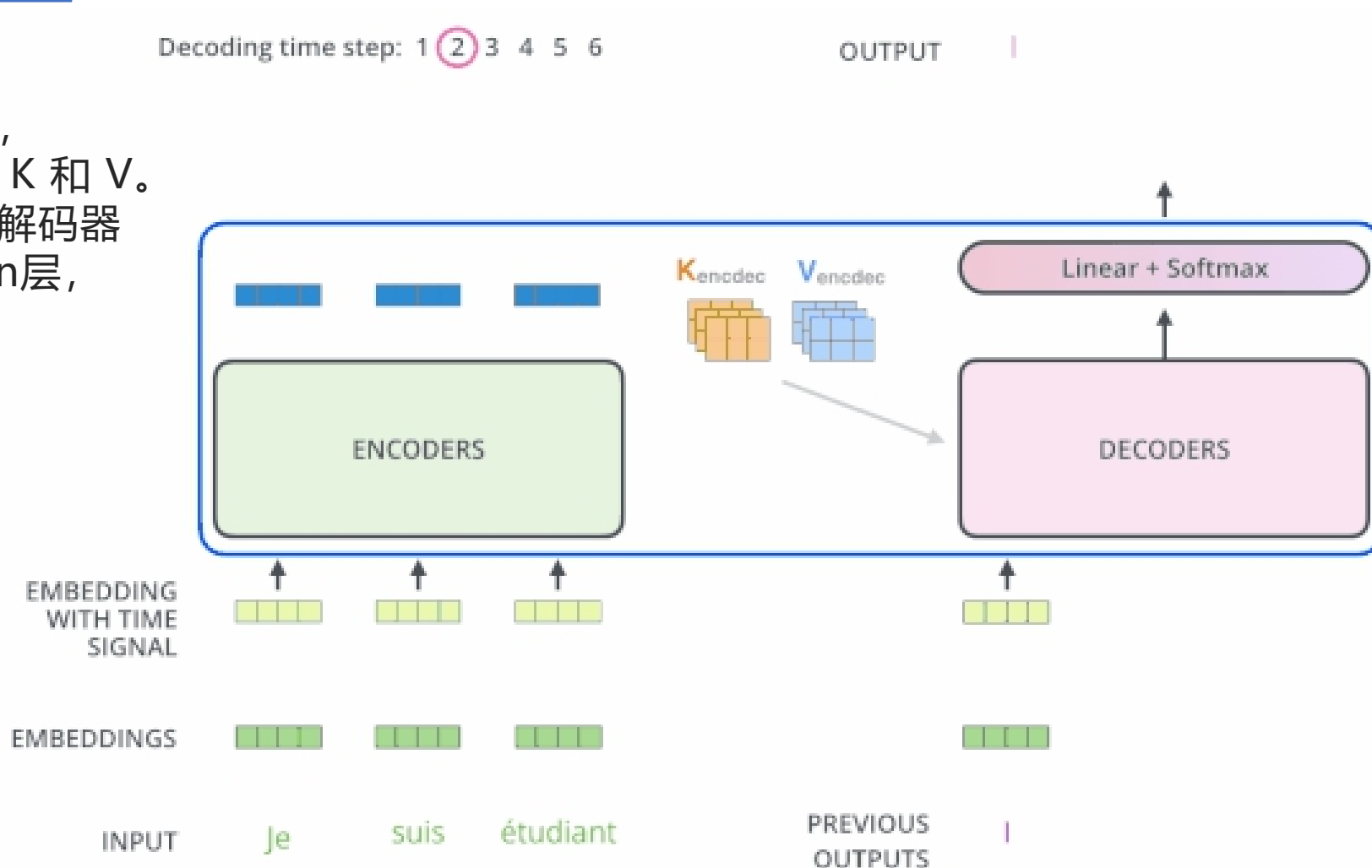
残差连接

编码器的每个子层
Self Attention 层和 FFNN
都有残差连接和层标准化。



解码器decoder

第一个编码器的输入是一个序列，
最后一个输出是一组注意力向量 K 和 V 。
这些注意力向量将会输入到每个解码器的
Encoder-Decoder Attention层，
这有助于解码器把注意力集中
输入序列的合适位置。



线性层和soft层

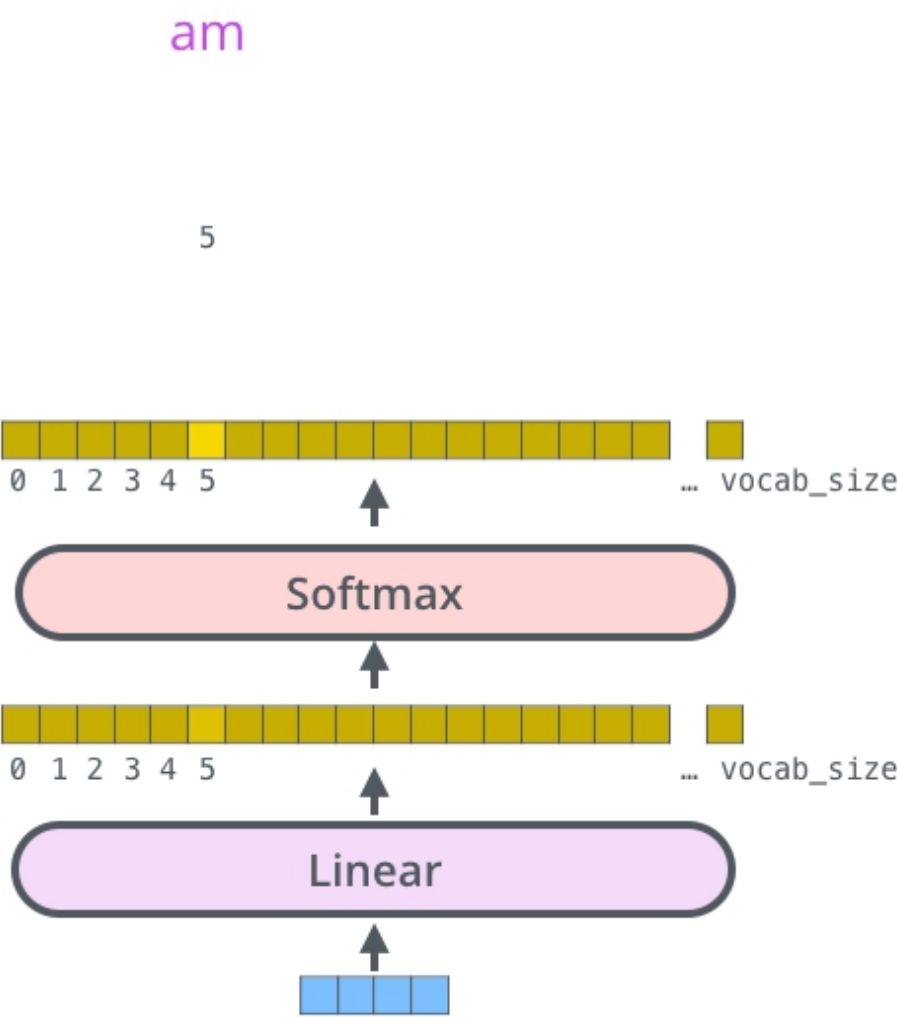
Which word in our vocabulary
is associated with this index?

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

把所有的分数转换为正数，并且加起来等于 1 **log_probs**

线性层就是一个普通的全连接神经网络
可以把解码器输出的向量，
映射到一个更长的向量，这个向量称为 logits 向量

Decoder stack output



实验结果

硬件参数：8个P100Gpu、每步0.4s，base model用了12个小时的训练时间。更大的模型则用3.5天。

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	41.29	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$3.3 \cdot 10^{18}$	
Transformer (big)	28.4	41.8	$2.3 \cdot 10^{19}$	

Faster Transformer

[编辑](#)[讨论](#)[上传视频](#)[收藏](#)

0



本词条缺少概述图，补充相关内容使词条更完整，还能快速升级，赶紧来[编辑](#)吧！

Faster Transformer ^[1] 是NVIDIA 针对Transformer推理提出的性能优化方案，这是一个BERT Transformer 单层前向计算的高效实现，代码简洁明了，后续可以通过简单修改支持多种Transformer结构。 ^[2]

Faster Transformer已经开源。 ^[2-3]