第七节 transformer框架

attention的优缺点

优点：

1. 解决了长序列依赖问题
2. 可以并行

缺点：

1. 开销变大了
2. 既然可以并行，也就是说，词与词之间不存在顺序关系（打乱一句话，这句话里的每个词的词向量依然不会变），即无位置关系（既然没有，我就加一个，通过位置编码的形式加）

一、位置编码（positional encoding）

为什么需要位置编码

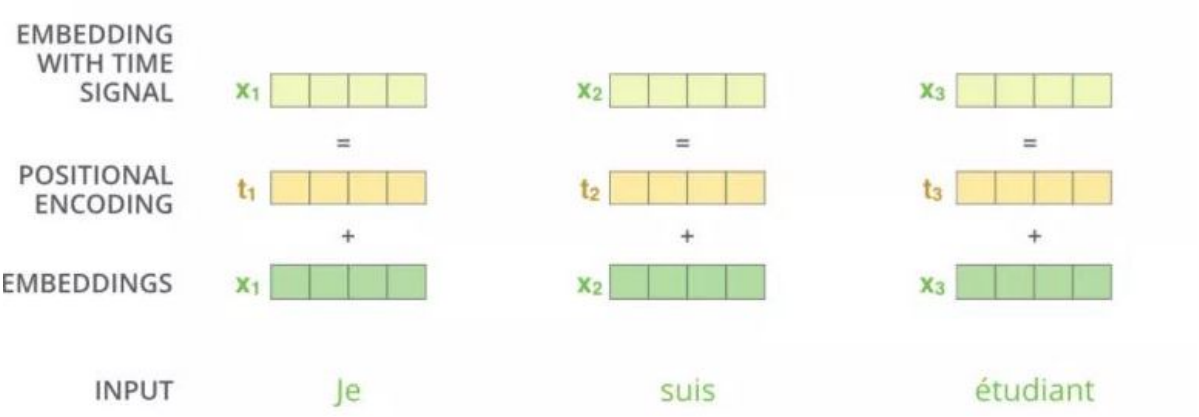
我们回忆一下之前的RNN和LSTM，它们在输入一句话的时候，其实是会按照先后顺序去输入的，所以它们在时序方面有着先天的优势，也就是它们自带了时序的特征。

但是在attention里面，我们可以看到，每个词的运算都是独立的，虽然这样可以并行提高效率，但是同时也忽略掉了很重要的时序信息，也就是不同的词的相对位置（或者理解为先后顺序也没问题）。

因此，位置编码的引入就是要给每个得到的词向量里面加入相对位置（或者先后顺序）的特征，让最终得到的词向量里面不仅包含词本身的含义，还包含它和其他词的相对位置。

位置编码是怎么做的

在原来attention得到的x1词向量的基础上加上位置编码得到一个新的x1

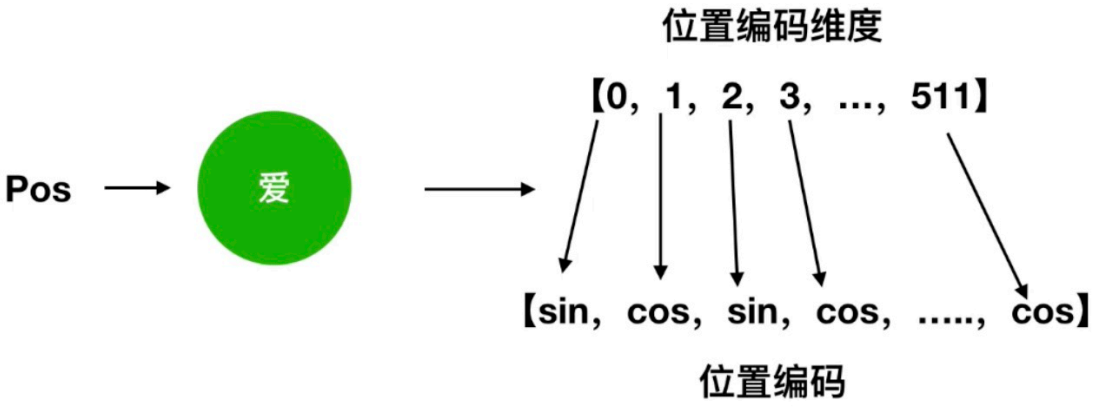


位置编码具体做法

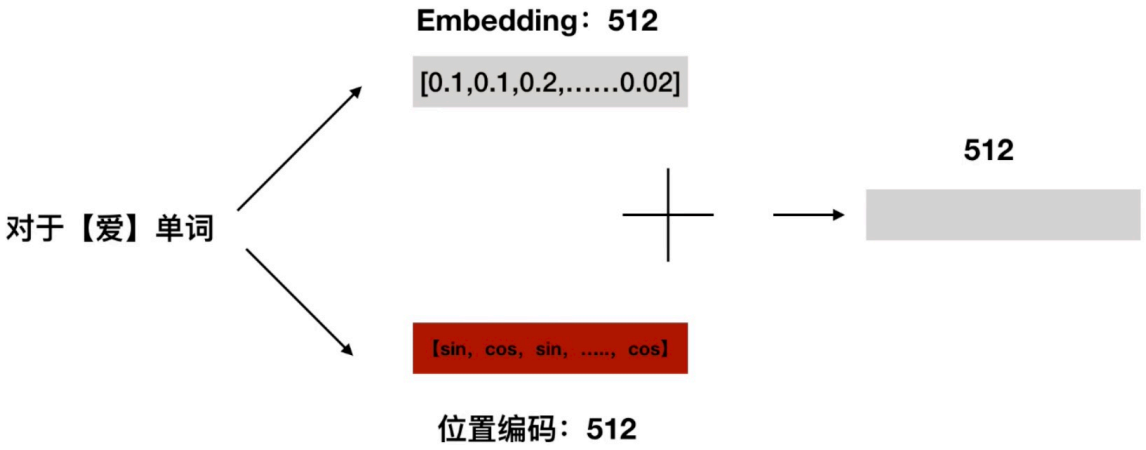
做法1：

这里的pos是位置，i是维度

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



做法2：



为什么这样做有用

pos+K=5，我在计算第 5 个单词的位置编码的时候

pos=1，k=4

pos=2，k=3

pos=3，k=2

pos=4，k=1

通过下面的公式，我们可以天然地就可以得到第五个单词和前面几个单词的相对位置信息（这里其实是利用了三角函数的积化和差性质）

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。

二、Transformer框架概述

1000*0.04=40-->10

5000*0.04=200-->20

预训练--》NNLM--》word2Vec--》ELMo--》Attention

NLP 中预训练的目的，其实就是为了生成词向量

顺水推舟，transformer 其实就是 attention 的一个堆叠

我们这里先从一个宏观的角度，去看 transformer 到底在干嘛，然后在细分，再作总结

其实，总体而言，transformer框架其实就是seq2seq

我们的输入是一句话，一个视频等等，在第一步的时候我们都要进行embedding，得到一个个序列，最后经过转换，我们最终会得到一个新的序列，或者可以称为向量，也就是序列（编码器）到序列（解码器）

整体分成两部分，编码器和解码器

整体框架

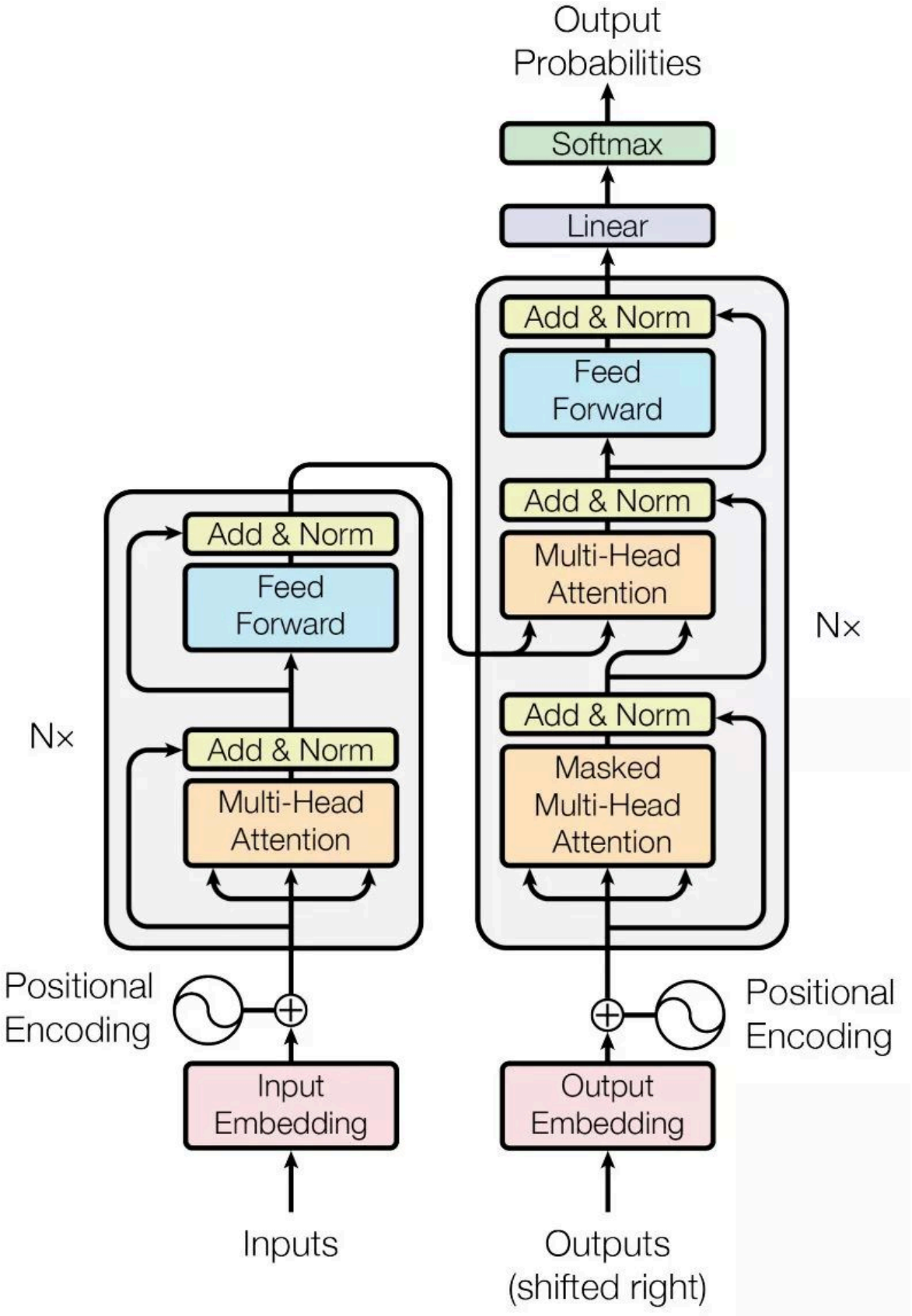


Figure 1: The Transformer - model architecture.

例子：机器翻译（by transformer）

通过机器翻译来做解释

给一个输入，给出一个输出（输出是输入的翻译的结果）

“我是一个学生” --》（通过 Transformer） I am a student

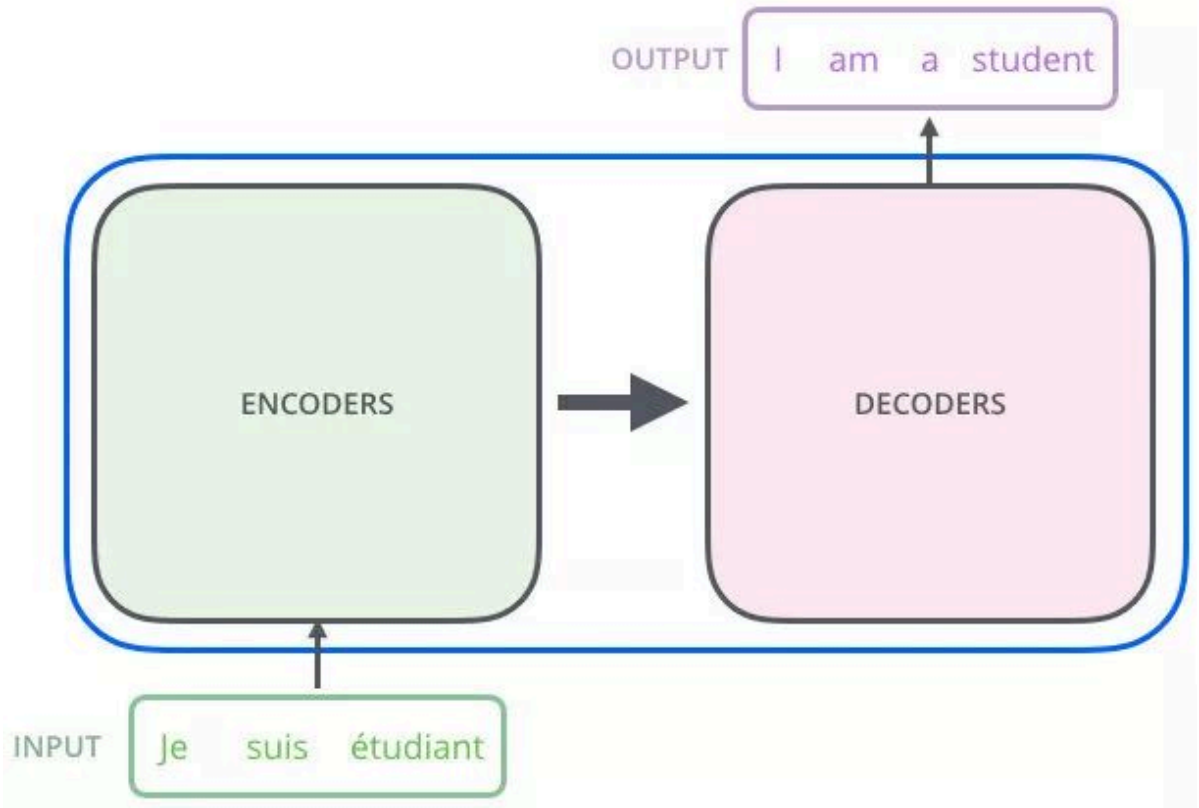
流程 1

编码器和解码器

编码器：把输入变成一个词向量（Self-Attetion）

解码器：得到编码器输出的词向量后，生成翻译的结果

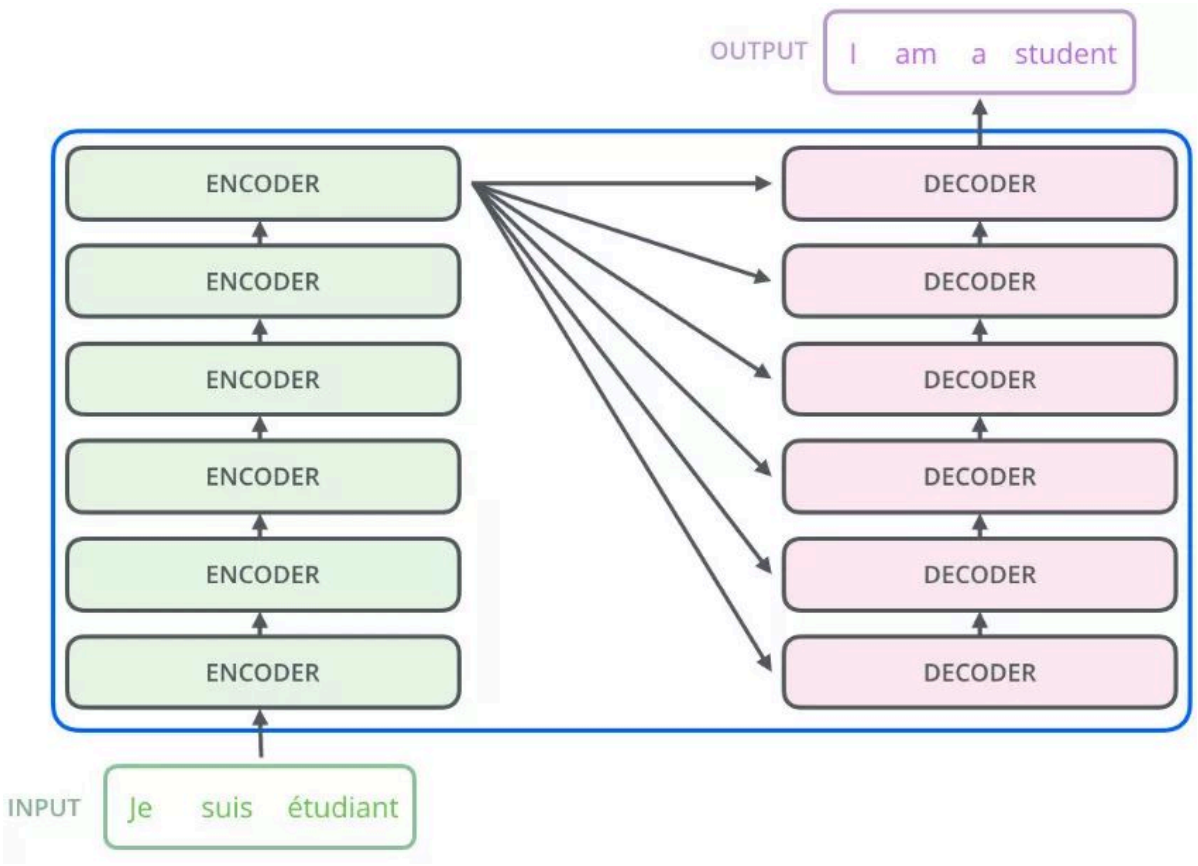
流程 2



Nx 的意思是，编码器里面又有 N 个小编码器（默认 N=6，这个是经过实验得出的比较好的参数，不同任务可能不同）

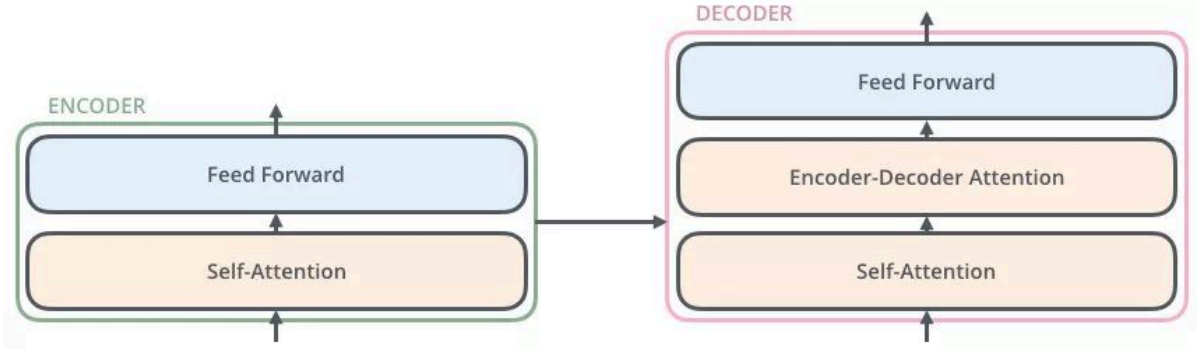
通过 6 个编码器，对词向量一步又一步的强化（增强）

流程 3



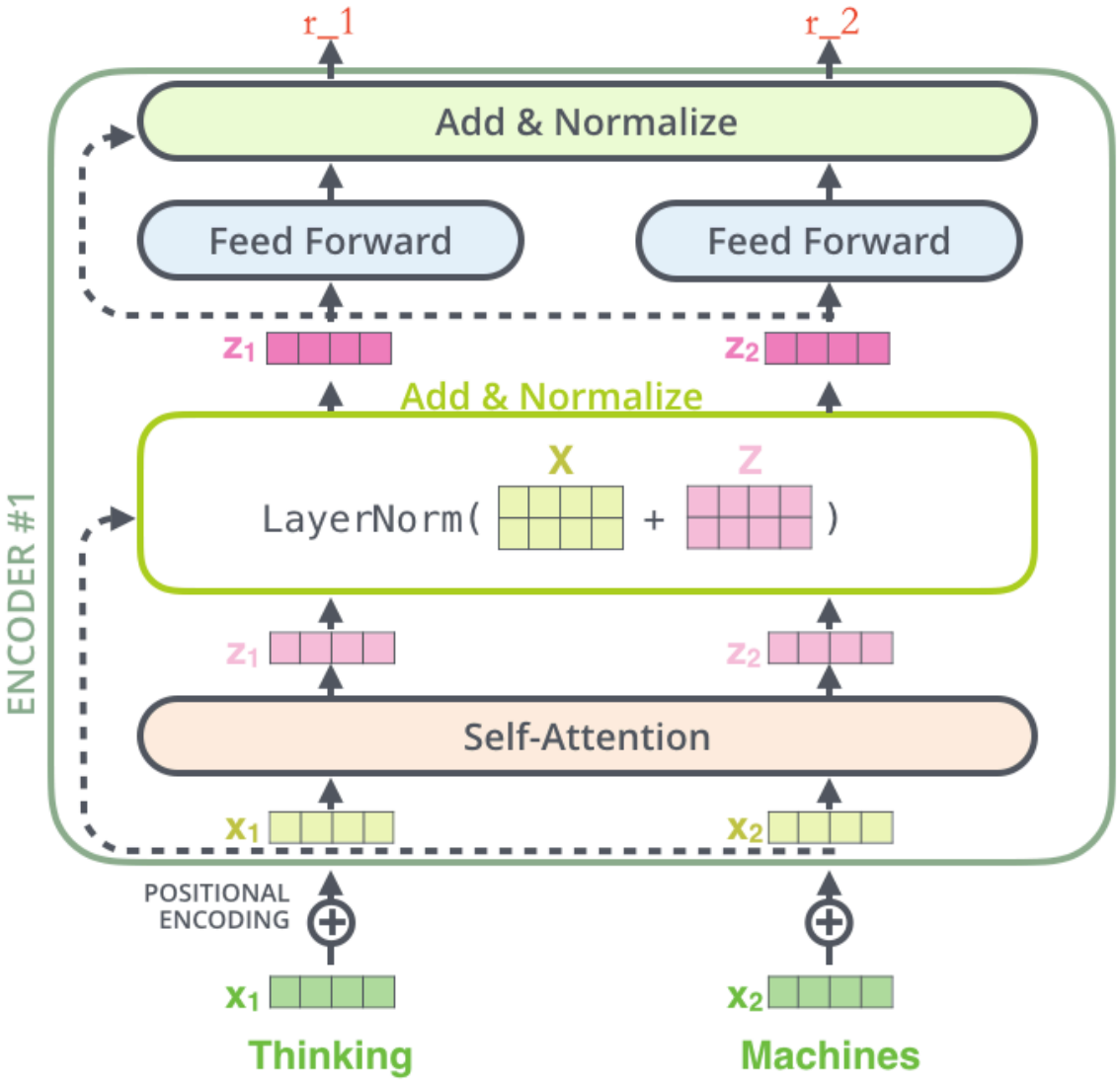
说了这么多，了解 Transformer 就是了解 Transformer 里的小的编码器（Encoder）和小的解码器（Decoder）

流程 4



FFN（Feed Forward，其实就是一个线性层）：w2（（w1x+b1））+b2

三、transformer的编码器（encoder）：我在做更好的词向量



Thinking

--》得到绿色的 x1（词向量，可以通过 one-hot、word2vec 得到）+ 叠加位置编码（给 x1 赋予位置属性）得到黄色的 x1

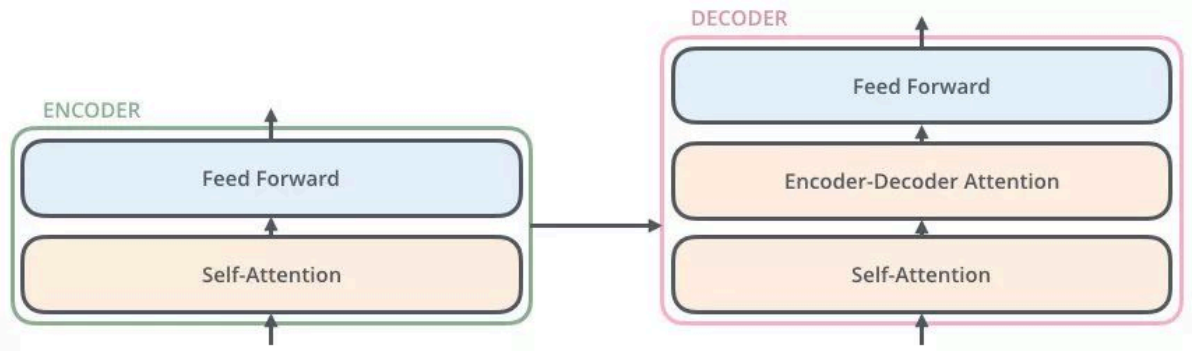
--》输入到 Self-Attention 子层中，做注意力机制（x1、x2 拼接起来的一句话做），得到 z1（x1 与 x1，x2拼接起来的句子做了自注意力机制的词向量，表征的仍然是 thinking），也就是说 z1 拥有了位置特征、句法特征、语义特征的词向量

--》残差网络（避免梯度消失，w3(w2(w1x+b1)+b2)+b3，如果 w1，w2，w3 特别小，0.0000000000000000……1，x 就没了，【w3(w2(w1x+b1)+b2)+b3+x】），归一化（LayerNorm），做标准化（避免梯度爆炸），得到了深粉色的 z1

--》Feed Forward，Relu（w2(w1x+b1)+b2），（前面每一步都在做线性变换，wx+b，线性变化的叠加永远都是线性变化（线性变化就是空间中平移和扩大缩小），通过 Feed Forward中的 Relu 做一次非线性变换，这样的空间变换可以无限拟合任何一种状态了），得到 r1（是 thinking 的新的表征）

总结下（这是重点，上面内容不用太过深究）：做词向量，只不过这个词向量更加优秀，让这个词向量能够更加精准的表示这个单词、这句话

四、transformer的解码器（decoder）



解码器的 Self-Attention 在编码已经生成的单词

假如目标词“我是一个学生”---》masked Self-Attention

训练阶段：目标词“我是一个学生”是已知的，然后 Self-Attention 是对“我是一个学生” 做计算

如果不做 masked，每次训练阶段，都会获得全部的信息

如果做 masked，Self-Attention 第一次对“我”做计算

Self-Attention 第二次对“我是”做计算

……

测试阶段：

1. 目标词未知，假设目标词是“我是一个学生”（未知），Self-Attention 第一次对“我”做计算
2. 第二次对“我是”做计算
3. ……

而测试阶段，没生成一点，获得一点

生成词

Linear 层转换成词表的维度

softmax 得到最大词的概率

Which word in our vocabulary is associated with this index?

am

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

5