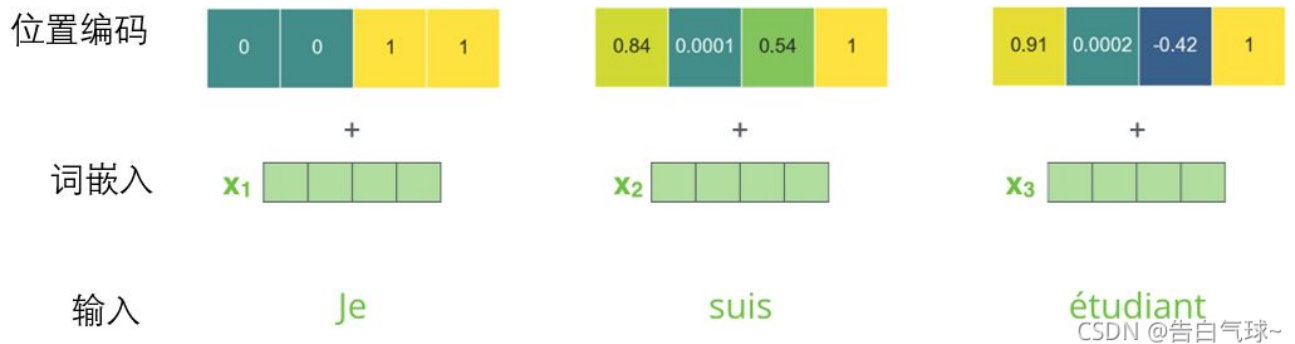


Day5: Transformer (下)

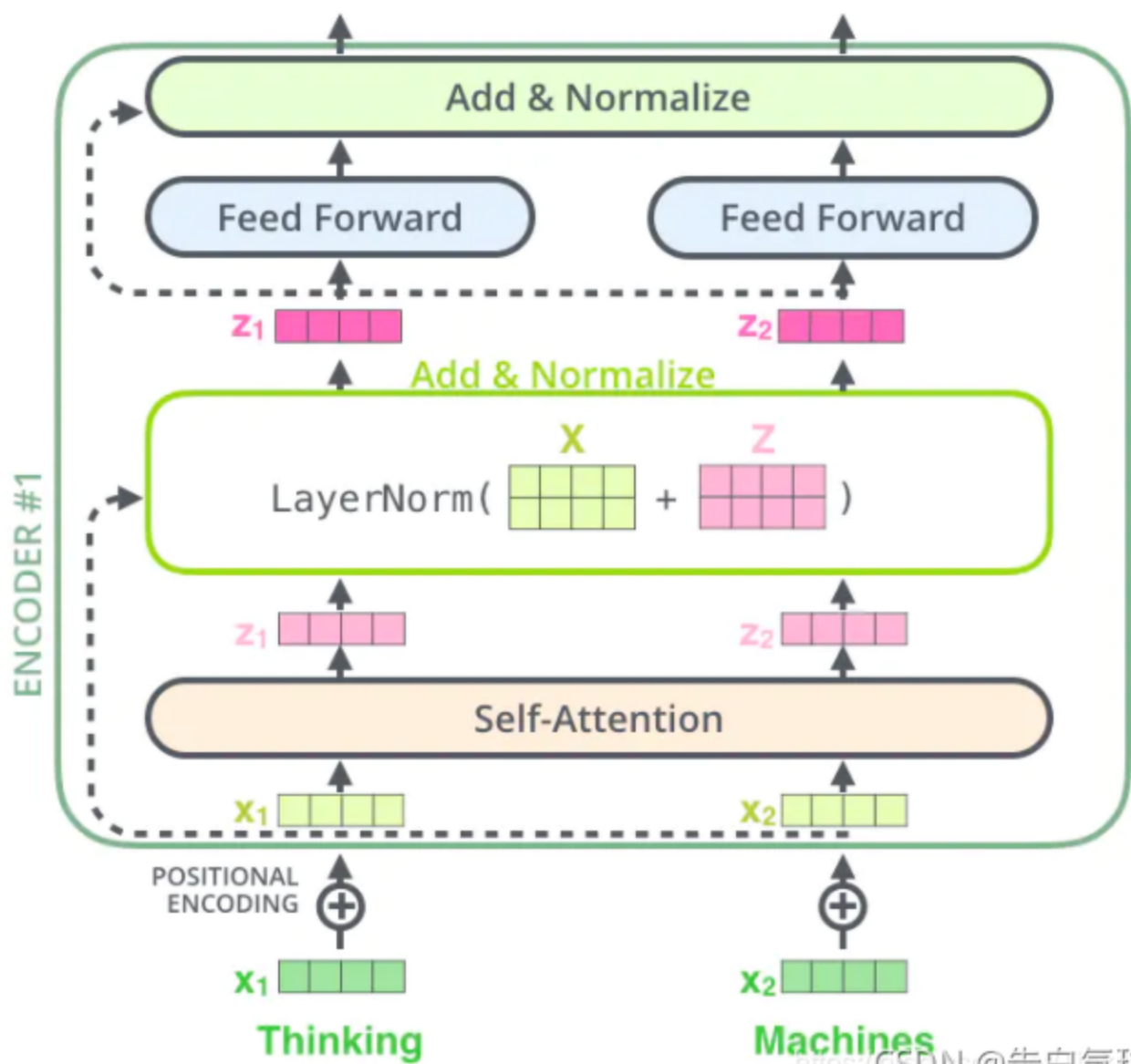
1. 补充昨天编码器的部分

为什么要用位置编码?

- 如果不添加位置编码，那么无论单词在什么位置，它的注意力分数都是确定的。这不是我们想要的。
- 为了理解单词顺序，Transformer为每个输入的词嵌入添加了一个向量，这样能够更好的表达词与词之间的关系。词嵌入与位置编码相加，而不是拼接，他们的效率差不多，但是拼接的话维度会变大，所以不考虑。




在经过多头注意力机制得到矩阵 Z 之后，并没有直接传入全连接神经网络，而是经过了一步Add&Normalize。



[@告白气球35](https://CSDN)

Add

就是在 Z 的基础上加了一个残差块 X  https://i-blog.csdnimg.cn/blog_migrate/b9951837dd639046ca37c9fba8b4efc5.png#pic_center
title="" alt="在这里插入图片描述" data-bbox="114 114 617 128"/> data-align="center">

为了了解残差块，我们引入ResNet残差神经网络，神经网络退化指的是在达到最优网络层数之后，神经网络还在继续训练导致Loss增大，对于多余的层，我们需要保证多出来的网络进行恒等映射。只有进行了恒等映射之后才能保证这多出来的神经网络不会影响到模型的效果。残差连接主要是为了防止网络退化。

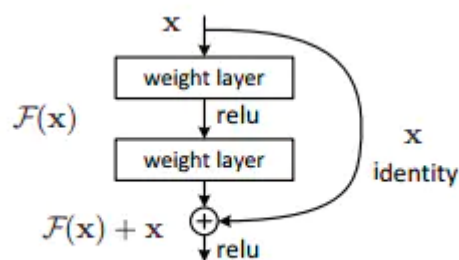


Figure 2. Residual learning: a building block.

上图就是构造的一个残差块， x 是输入值， $F(x)$ 是经过第一层线性变换后并且激活的输出，在第二层线性变化之后，激活之前， $F(x)$ 加入了这一层输入值 x ，然后再进行激活后输出。要恒等映射，我们只需要让 $F(x) = 0$ 就可以了。 x 经过线性变换（随机初始化权重一般偏向于0），输出值明显会偏向于0，而且经过激活函数Relu会将负数变为0，过滤了负数的影响。这样当网络自己决定哪些网络层为冗余层时，使用ResNet的网络很大程度上解决了学习恒等映射的问题，用学习残差 $F(x)=0$ 更新该冗余层的参数来代替学习 $h(x)=x$ 更新冗余层的参数。

Normalize

归一化目的：

- 1、加快训练速度
- 2、提高训练的稳定性

使用到的归一化方法是Layer Normalization。

计算公式为：

$$LayerNorm(X + MultiHeadAttention(X))$$

$$LayerNorm(X + FeedForward(X))$$

2. 编码器结构

Decoder也是由6个decoder堆叠而成的（ $N=6$ ）。包含两个 Multi-Head Attention 层。第一个 Multi-Head Attention 层采用了 Masked 操作。第二个 Multi-Head Attention 层的 K, V 矩阵使

用 Encoder 的编码信息矩阵C进行计算，而Q使用上一个 Decoder block 的输出计算。

Masked Multi-Head Attention

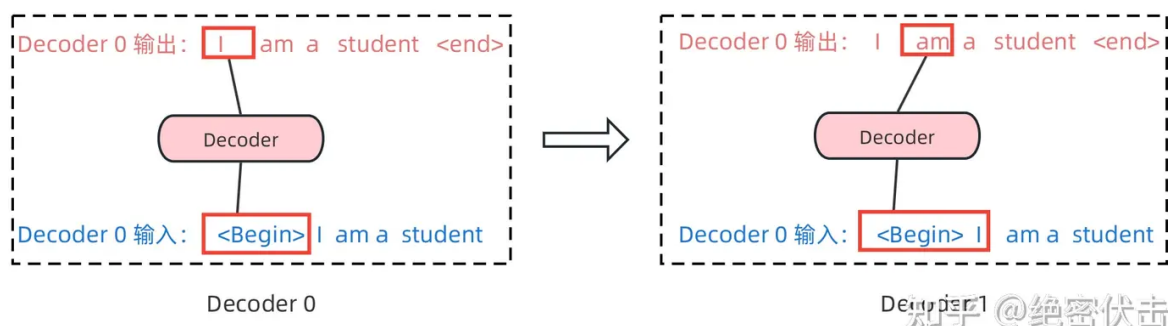
与Encoder的Multi-Head Attention计算原理一样，只是多加了一个mask码。mask 表示掩码，它对某些值进行掩盖，使其在参数更新时不产生效果。Transformer 模型里面涉及两种 mask，分别是 padding mask 和 sequence mask。

padding mask

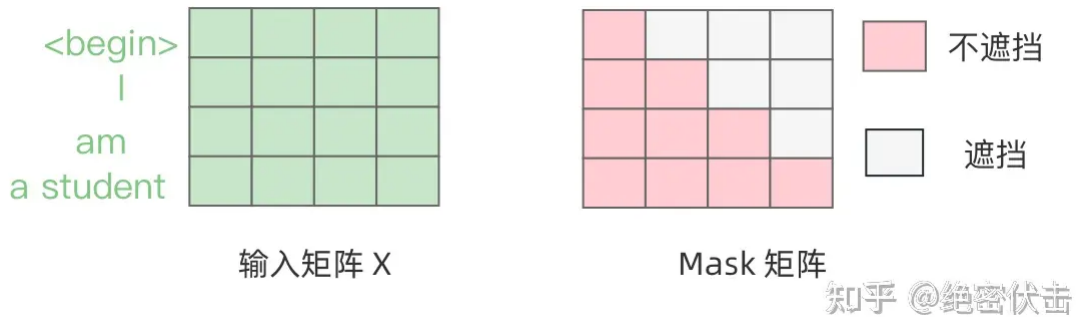
因为每个批次输入序列长度是不一样的也就是说，我们要对输入序列进行对齐。具体来说，就是给在较短的序列后面填充 0。但是如果输入的序列太长，则是截取左边的内容，把多余的直接舍弃。因为这些填充的位置，其实是没什么意义的，所以我们的 attention 机制不应该把注意力放在这些位置上，所以我们需要进行一些处理。具体的做法是，把这些位置的值加上一个非常大的负数(负无穷)，这样的话，经过 softmax，这些位置的概率就会接近0。

sequence mask

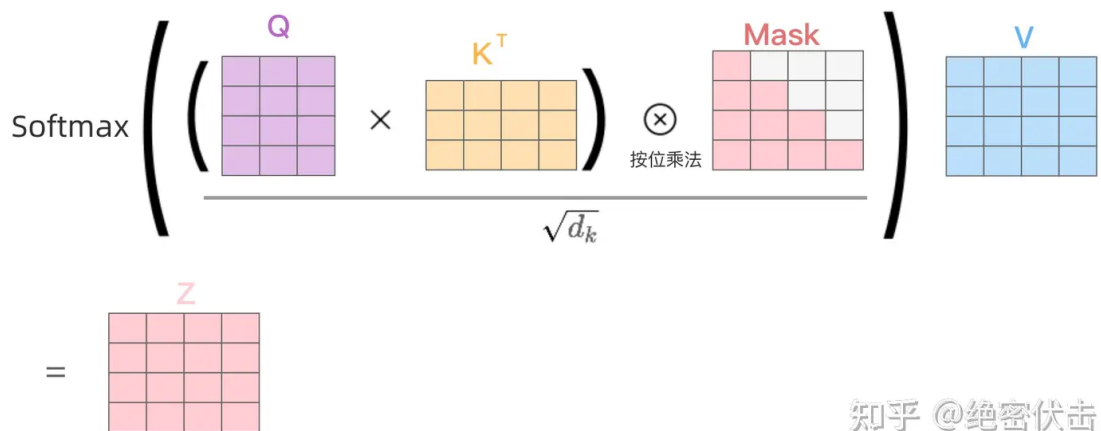
sequence mask 是为了使得 decoder 不能看见未来的信息。对于一个序列，在 time_step 为 t 的时刻，我们的解码输出应该只能依赖于 t 时刻之前的输出，而不能依赖 t 之后的输出。那么具体怎么做法：产生一个上三角矩阵，上三角的值全为0。把这个矩阵作用在每一个序列上，就可以达到我们的目的。



step1: 输入矩阵包含"I am a student"4个单词的表示向量, **Mask**是一个 4×4 的矩阵。在**Mask**可以发现单词""只能使用单词""的信息, 而单词"I"可以使用单词"I"的信息, 即只能使用之前的信息。



step2: 接下来的操作和之前Encoder中的Self-Attention一样, 只是在Softmax之前需要进行Mask操作。



step3: 通过上述步骤就可以得到一个Mask Self-Attention的输出矩阵 Z_i , 然后和Encoder类似, 通过Multi-Head Attention拼接多个输出 Z_i , Z 与输入 X 维度一样。

3. 输出

Output如图中所示, 首先经过一次线性变换(线性变换层是一个简单的全连接神经网络, 它可以把解码组件产生的向量投射到一个比它大得多的, 被称为对数几率的向量里), 然后Softmax得到输出的概率分布(softmax层会把向量变成概率), 然后通过词典, 输出概率最大的对应的单词作为我们的预测输出。

总结

优点：效果好、可以并行训练，速度快、很好的解决了长距离依赖的问题 缺点：完全基于self-attention，对于词语位置之间的信息有一定的丢失，虽然加入了positional encoding来解决这个问题，但也还存在着可以优化的地方。