Transformer

前言

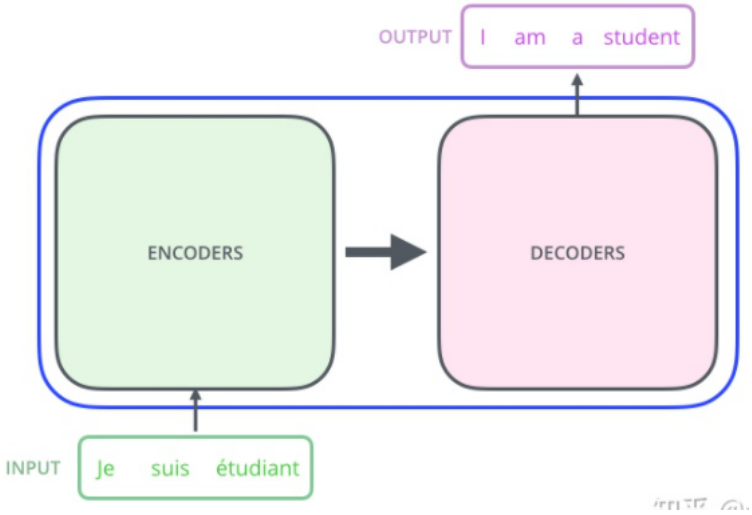
- Transformer由`self-Attention`和`Feed Forward Neural Network`组成。一个基于Transformer的可训练的神经网络可以通过堆叠Transformer的形式进行搭建。
- 之前RNN的计算是顺序的，只能从左向右或从右向左依次计算，这种机制带来两个问题：
  1. 时间片t的计算依赖于t-1时刻的计算结果，这样限制了模型的并行能力。
  2. 顺序计算的过程中信息会丢失，尽管LSTM等机制的结构在一定程度上缓解了长期依赖的问题，但是对于特别长期的依赖现象，LSTM依旧无能为力。
- Transformer首先使用Attention机制，将序列中的任意两个位置之间的距离缩小为一个常量；其次不是类似RNN的顺序结构，因此具有更好的并行性。

1. Tranformer详解

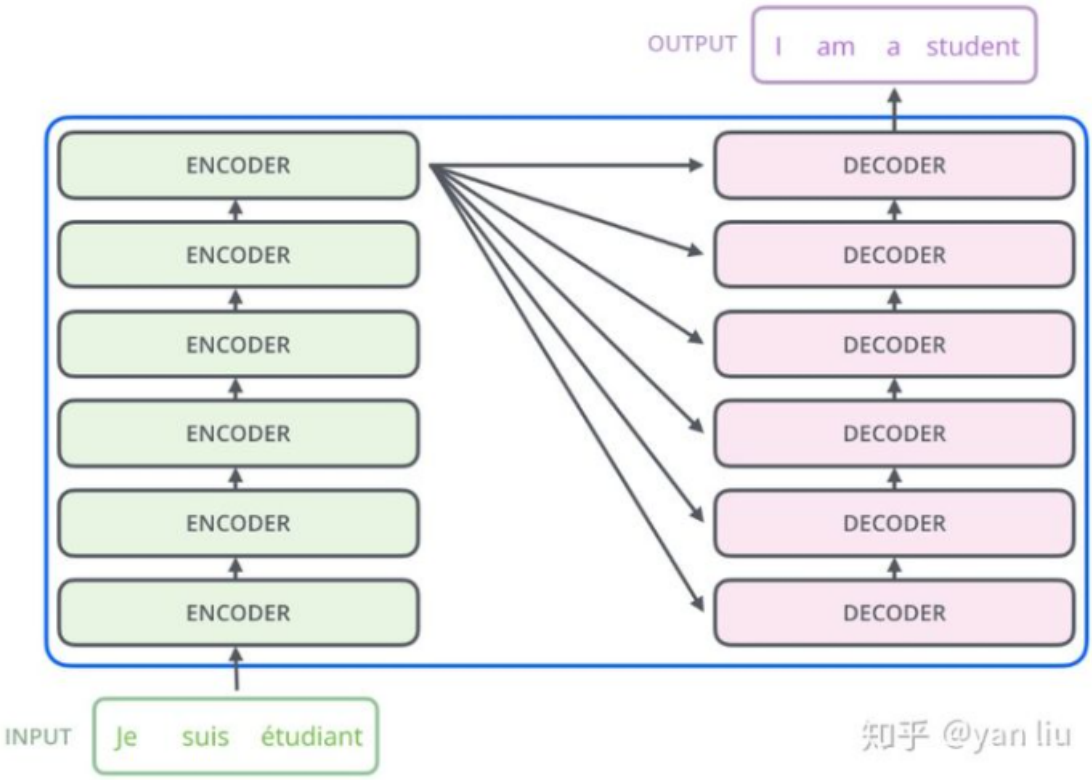
1.1 高层Transformer

- Transformer用于机器翻译

- Transformer的本质是一个Encoder-Decoder的结构，上图可以表示为：



- 编码器由6个block组成，解码器是6个解码block。编码器的输出会作为解码器的输入。



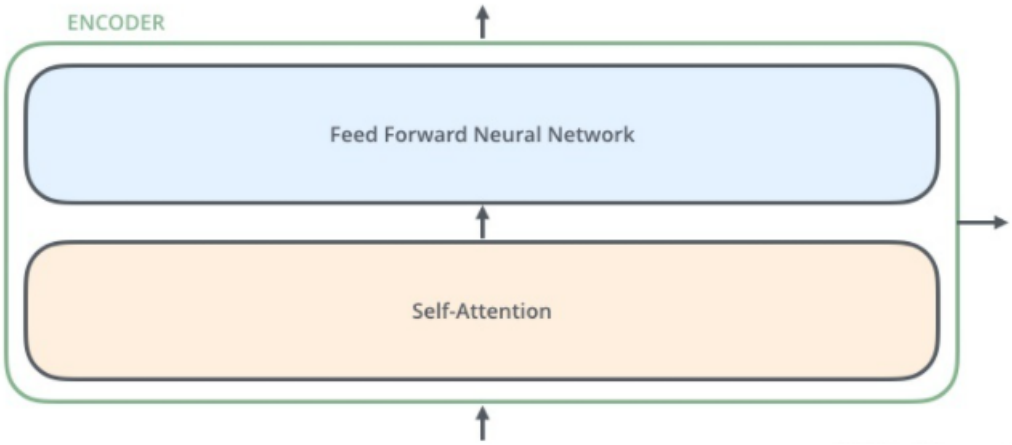
在Transformer的encoder中，数据首先会经过`self-attention`的模块得到一个加权之后的特征向量Z，这个`Z`也就是下式中的`Attention(Q,K,V)`：

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

得到Z之后，它会被送到encoder的下一个模块，即Feed Forward Neural Network。这个全连接有两层，第一层的激活函数是ReLU，第二层是一个线性激活函数，可以表示为：

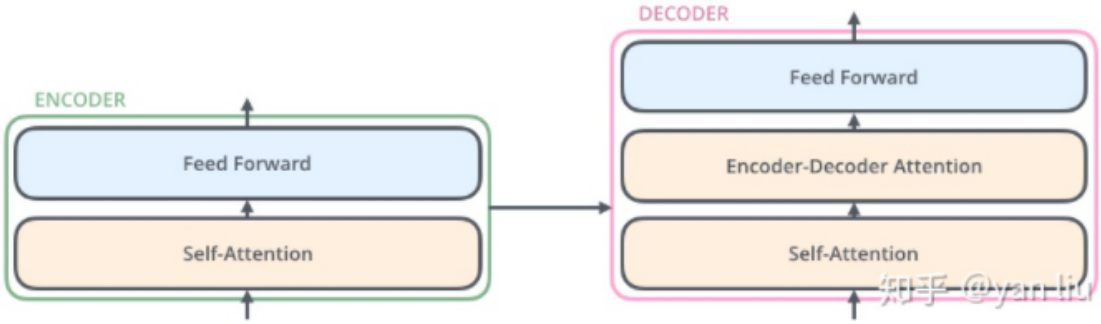
$$\text{FFN}(Z)=max(0,ZW_1+b_1)W_2+b_2$$

Encoder的结构如图：



Decoder和Encoder的不同之处在于Decoder多了一个Encoder-Decoder Attention，两个Attention分别用于计算输入和输出的权值：

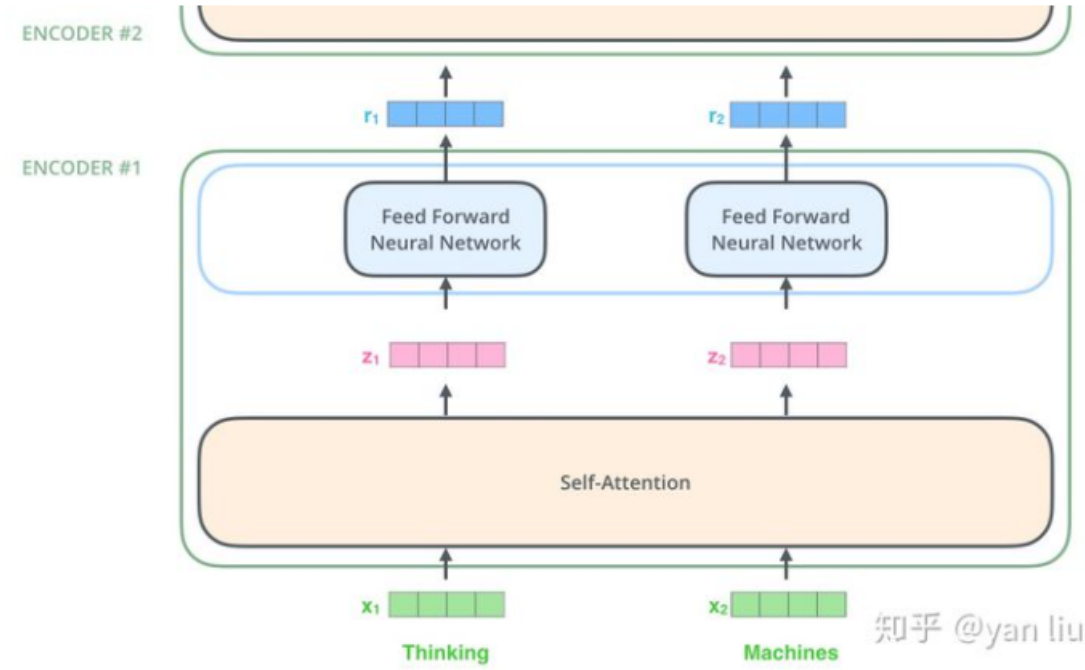
- Self-Attention：当前翻译和已经翻译的前文之间的关系。
- Encoder-Decoder Attention：当前翻译和编码的特征向量之间的关系。



1.2 输入编码

首先通过Word2Vec等词嵌入方法将输入语料库转化成特征向量。

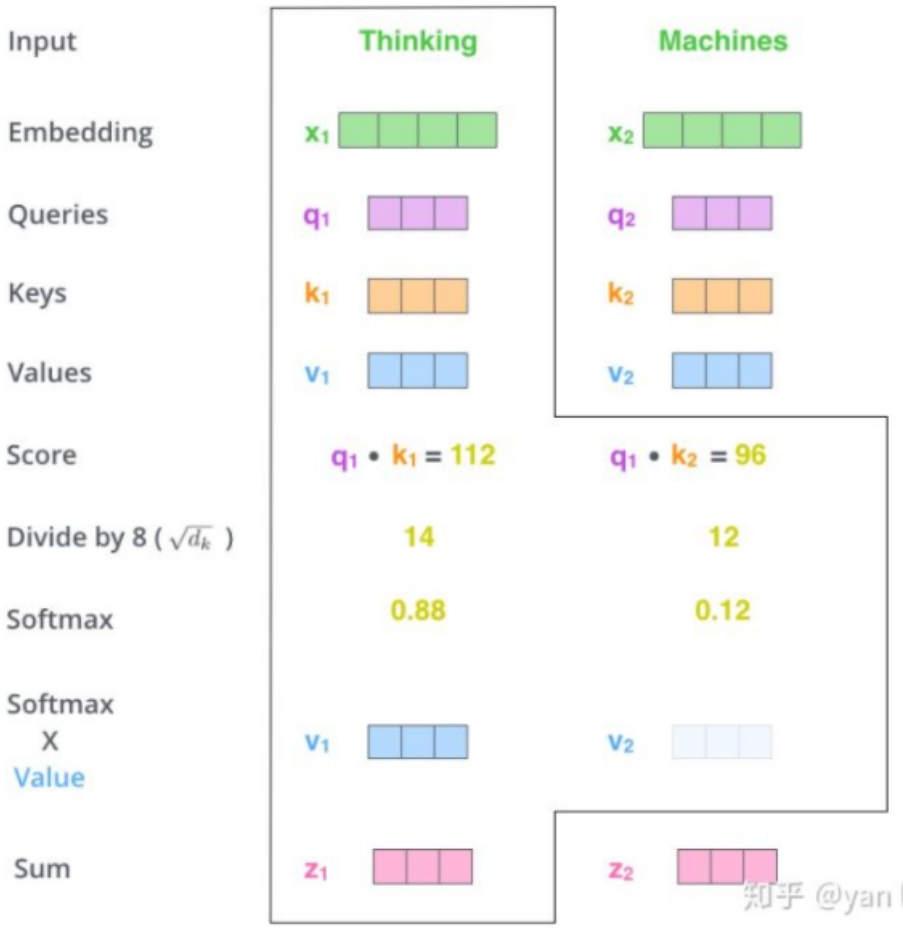
在最底层的block中，x直接作为transformer的输入，而在其他层中，输入则是上一个block的输出。



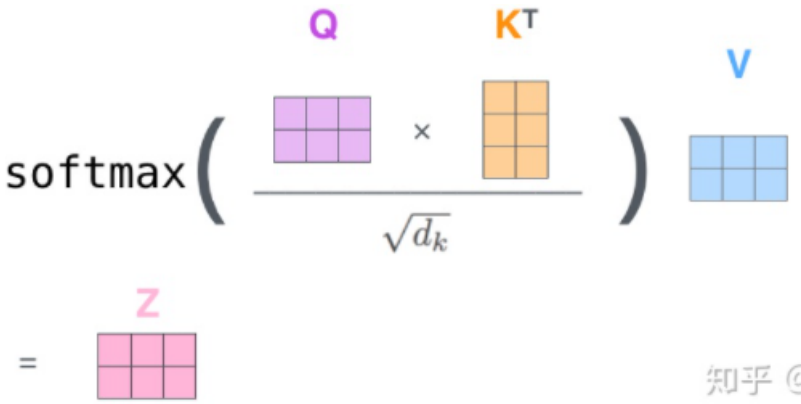
1.3 Self-Attention

整个过程分为7步：

1. 将输入单词转化成嵌入向量；
2. 根据嵌入向量得到`q,k,v`三个向量；
3. 为每个向量计算一个score：$\text{score} = q \cdot k$
4. 为了梯度的稳定，Transformer使用了score归一化，即除以$\sqrt{d_k}$
5. 对score施以softmax激活函数；
6. softmax点乘Value值v，得到加权的每个输入向量的评分v；
7. 相加之后得到最终的输出结果z：$z = \sum v$

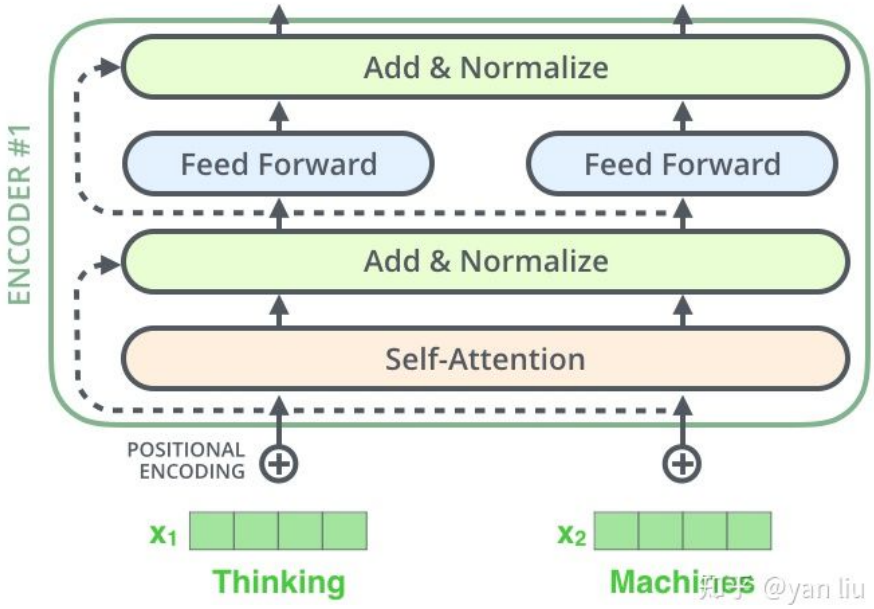


上图可以总结为如图的矩阵形式：



这就是上述公式`Attention(Q,K,V)`的计算方式。

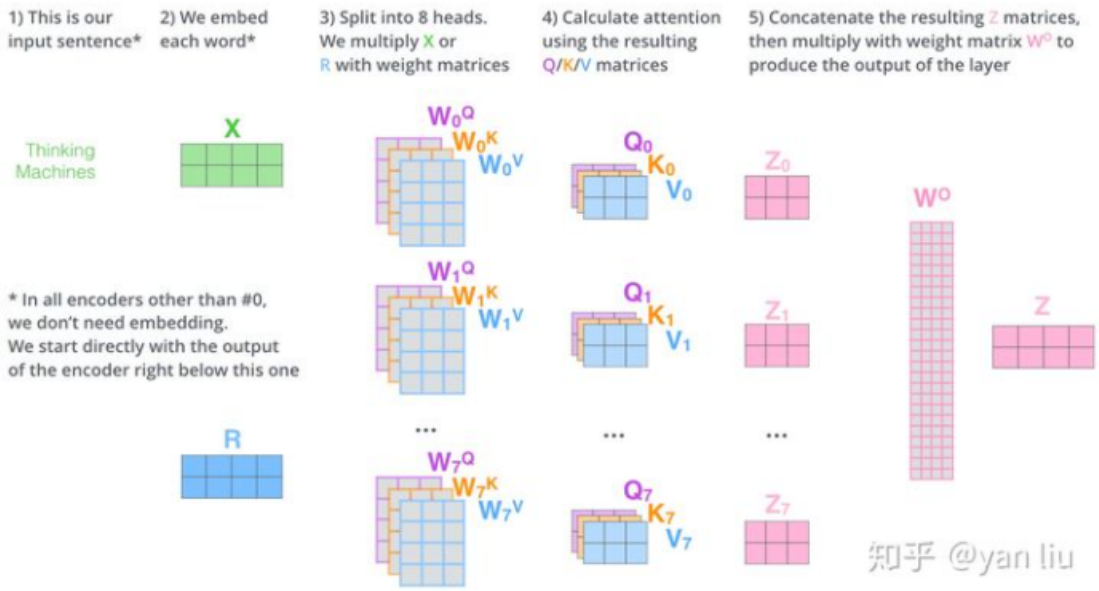
self-attention采用了残差网络中的short-cut结构，解决深度学习中的退化问题。



1.3 Multi-Head Attention

Multi-Head Attention相当于h个不同的self-attention的集成，输出分成3步：

1. 将数据X分别输入到上图的h个self-attention中，得到h个加权后的特征矩阵Zi。
2. 将h个Zi按列拼成一个大的特征矩阵。
3. 特征矩阵经过一层全连接后得到输出Z。



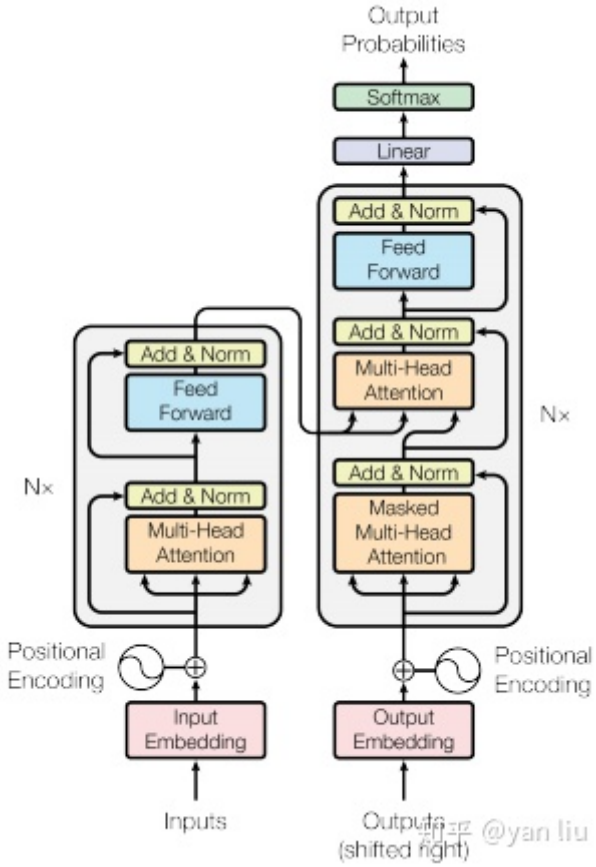
1.5 Encoder-Decoder Attention

在encoder-decoder attention中，Q来自于解码器的上一个输出，K和V来自于解码器的输出。过程和self-attention一样。

1.5 损失层

解码器解码之后，解码的特征向量经过一层激活函数为softmax的全连接层之后得到反映每个单词概率的输出向量，此时可以通过CTC等损失函数训练模型了。

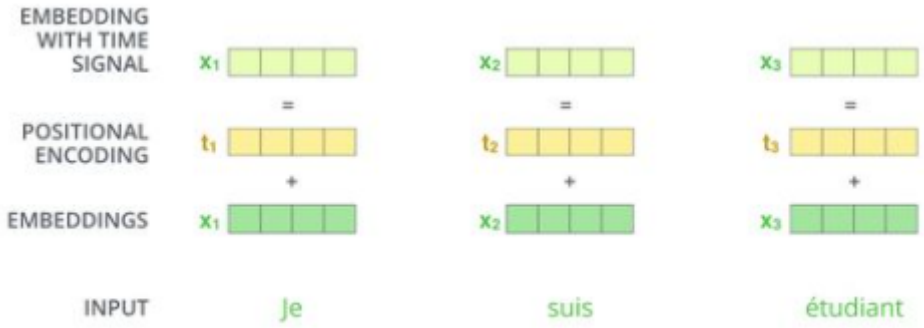
Transformer的完整结构图：



2. 位置编码

Transformer模型没有捕捉顺序序列的能力，无论句子的结构怎么打乱，都会得到同样的结果。为了解决这个问题，在编码词向量时引入了位置编码（Position Encoding）的特征。位置编码会在词向量中加入单词的位置信息，这样Transformer就可以区分不同位置的单词了。

位置编码的模式有：a. 根据数据学习；b. 自己设计编码规则；



编码公式：

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

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

`pos` 表示单词的位置，`i` 表示单词的维度。

3. 总结

优点：（1）Transformer是一个全连接加Attention的结合体，抛弃了RNN和CNN，取得了不错的效果。（2）Transformer的设计带来最大性能提升的关键是任意两个单词的距离是1，解决了NLP中的长期依赖问题。（3）算法的并行性非常好。

缺点：（1）使模型丧失了捕捉局部特征的能力。（2）Transformer失去的位置信息在NLP中非常重要，加入的位置编码并没有改变Transformer结构上的固有缺陷。