详解Transformer （Attention Is All You Need）

刘岩
算法菜鸡

前言

注意力（Attention）机制[2]由Bengio团队与2014年提出并在近年广泛的应用在深度学习中的各个领域，例如在计算机视觉方向用于捕捉图像上的感受野，或者NLP中用于定位关键token或者特征。谷歌团队近期提出的用于生成词向量的BERT[3]算法在NLP的11项任务中取得了效果的大幅提升，堪称2018年深度学习领域最振奋人心的消息。而BERT算法的最重要的部分便是本文中提出的Transformer的概念。

正如论文的题目所说的，Transformer中抛弃了传统的CNN和RNN，整个网络结构完全是由Attention机制组成。更准确地讲，Transformer由且仅由self-Attenion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，作者的实验是通过搭建编码器和解码器各6层，总共12层的Encoder-Decoder，并在机器翻译中取得了BLEU值得新高。

作者采用Attention机制的原因是考虑到RNN（或者LSTM，GRU等）的计算限制为是顺序的，也就是说RNN相关算法只能从左向右依次计算或者从右向左依次计算，这种机制带来了两个问题：

1. 时间片 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，但是对于特别长期的依赖现象,LSTM依旧无能为力。

Transformer的提出解决了上面两个问题，首先它使用了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；其次它不是类似RNN的顺序结构，因此具有更好的并行性，符合现有的GPU框架。论文中给出Transformer的定义是：Transformer is the first transduction model relying entirely on self-attention to compute representations of its input and output without using sequence aligned RNNs or convolution.

遗憾的是，作者的论文比较难懂，尤其是Transformer的结构细节和实现方式并没有解释清楚。尤其是论文中的 Q ， V ， K 究竟代表什么意思作者并没有详细说明。通过查阅资料，发现了一篇非常优秀的讲解Transformer的技术博客[4]。本文中的大量插图也会从该博客中截取。首先感谢Jay Alammer详细的讲解，其次推荐大家去阅读原汁原味的文章。

1. Transformer 详解

1.1 高层Transformer

论文中的验证Transformer的实验室基于机器翻译的，下面我们就以机器翻译为例子详细剖析Transformer的结构，在机器翻译中，Transformer可概括为如图1：

图1：Transformer用于机器翻译

Transformer的本质上是一个Encoder-Decoder的结构，那么图1可以表示为图2的结构：

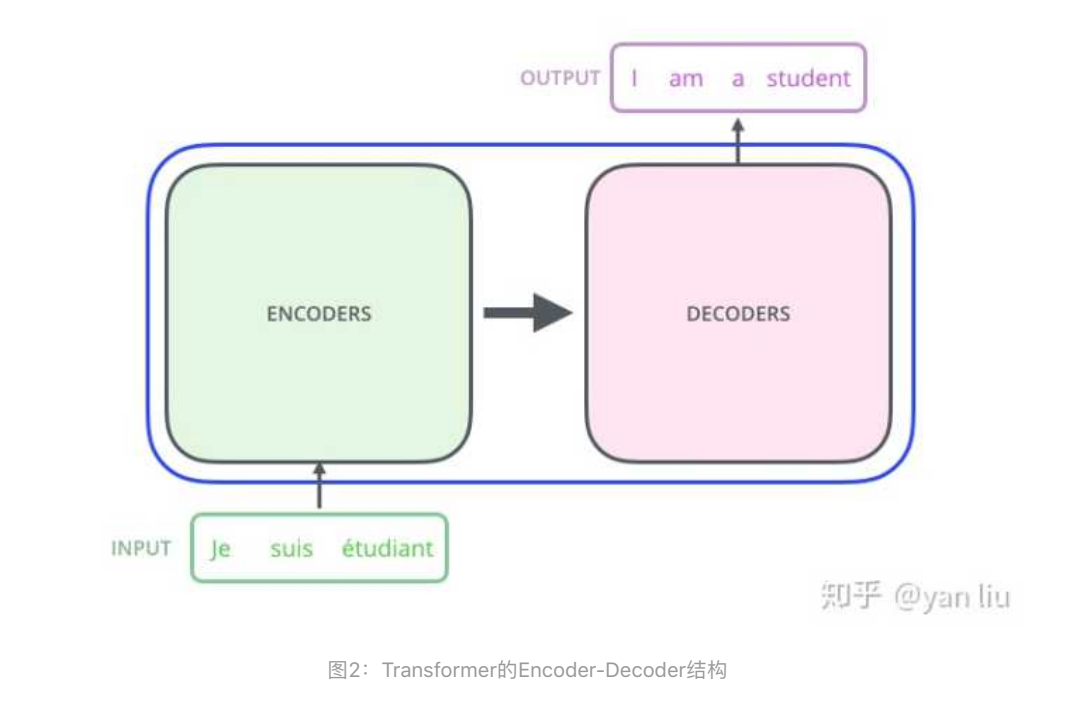

图2：Transformer的Encoder-Decoder结构

如论文中所设置的，编码器由6个编码block组成，同样解码器是6个解码block组成。与所有的生成模型相同的是，编码器的输出会作为解码器的输入，如图3所示：

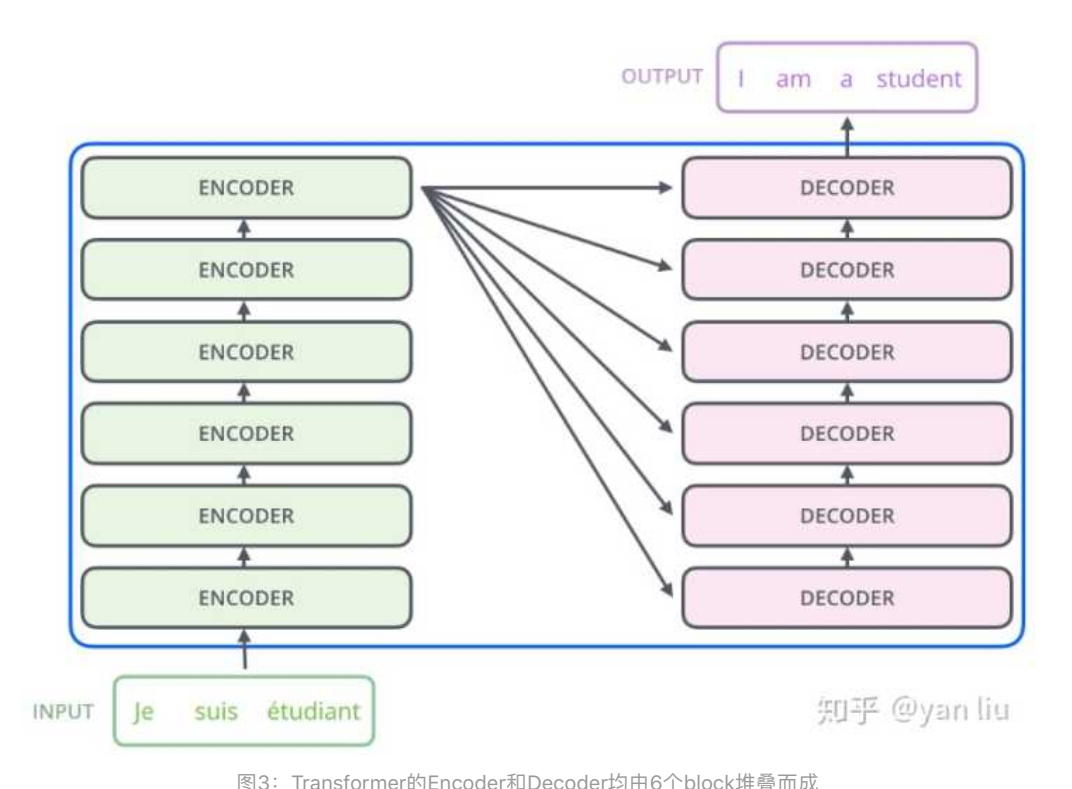

图3：Transformer的Encoder和Decoder均由6个block堆叠而成

我们继续分析每个encoder的详细结构：在Transformer的encoder中，数据首先会经过一个叫做'self-attention'的模块得到一个加权之后的特征向量 Z ，这个 Z 便是论文公式1中的 $\text{Attention}(Q,K,V)$ ：

$$\text{Attention}(Q,K,V)=\text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \quad (1)$$

第一次看到这个公式你可能会一头雾水，在后面的文章中我们会揭开这个公式背后的实际含义，在这一段暂时将其叫做 Z 。

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：

$$\text{FFN}(Z)=max(0,ZW_1+b_1)W_2+b_2 \quad (2)$$

Encoder的结构如图4所示：

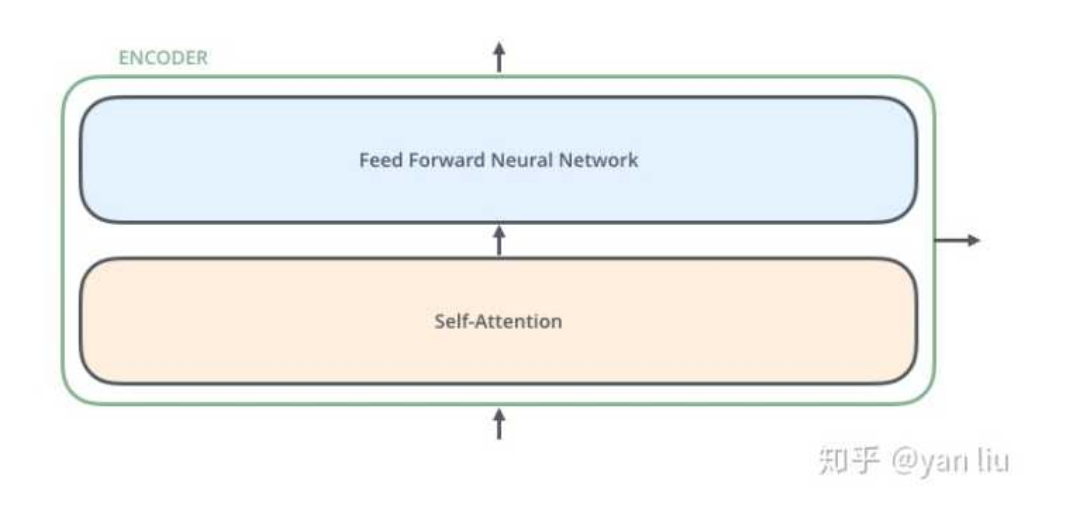

图4：Transformer由self-attention和Feed Forward neural network组成

Decoder的结构如图5所示，它和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：

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

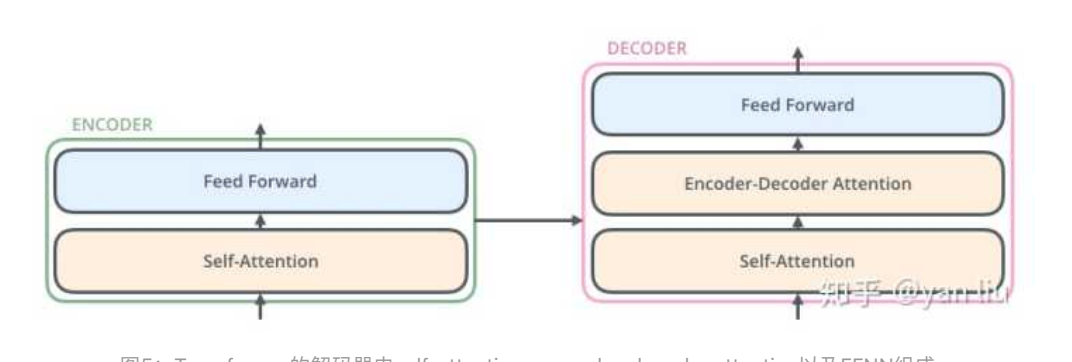

图5：Transformer的解码器由self-attention，encoder-decoder attention以及FFNN组成

1.2 输入编码

1.1节介绍的就是Transformer的主要框架，下面我们将介绍它的输入数据。如图6所示，首先通过Word2Vec等词嵌入方法将输入语料转化成特征向量，论文中使用的词嵌入的维度为 $d_{model}=512$ 。

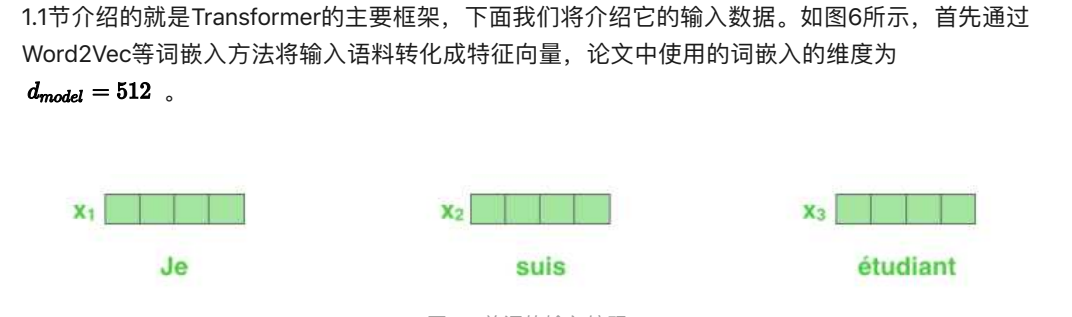

图6：单词的输入编码

在最底层的block中， x 将直接作为Transformer的输入，而在其他层中，输入则是上一个block的输出。为了画图更简单，我们使用更简单的例子来表示接下来的过程，如图7所示：

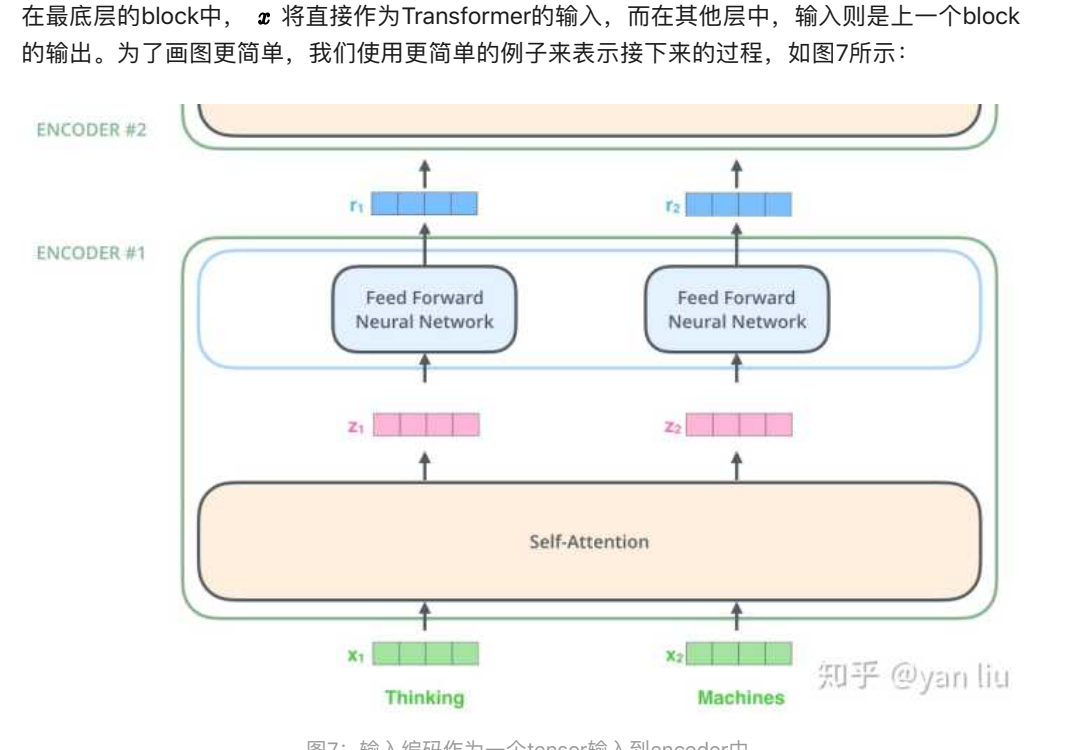

图7：输入编码作为一个tensor输入到encoder中

1.3 Self-Attention

Self-Attention是Transformer最核心的内容，然而作者并没有详细讲解，下面我们来补充一下作者遗漏的地方。回想Bahdanau等人提出的用Attention[2]，其核心内容是为输入向量的每个单词学习一个权重，例如在下面的例子中我们判断it代指的内容。

```
The animal didn't cross the street because it was too tired
```

通过加权之后可以得到类似图8的加权情况，在讲解self-attention的时候我们也会使用图8类似的表示方式

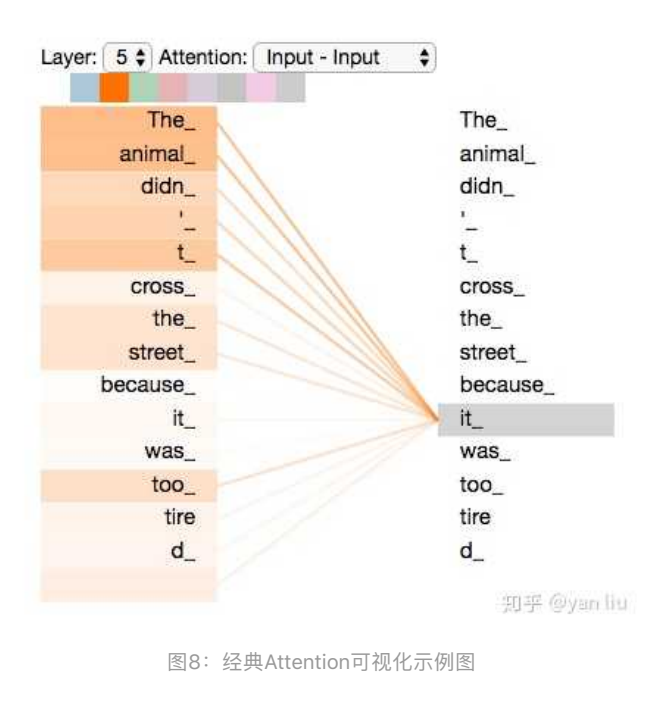

图8：经典Attention可视化示例图

在self-attention中，每个单词有3个不同的向量，它们分别是Query向量（ Q ），Key向量（ K ）和Value向量（ V ），长度均是64。它们是通过3个不同的权值矩阵由嵌入向量 X 乘以三个不同的权值矩阵 W^Q ， W^K ， W^V 得到，其中三个矩阵的尺寸也是相同的。均是 512×64 。

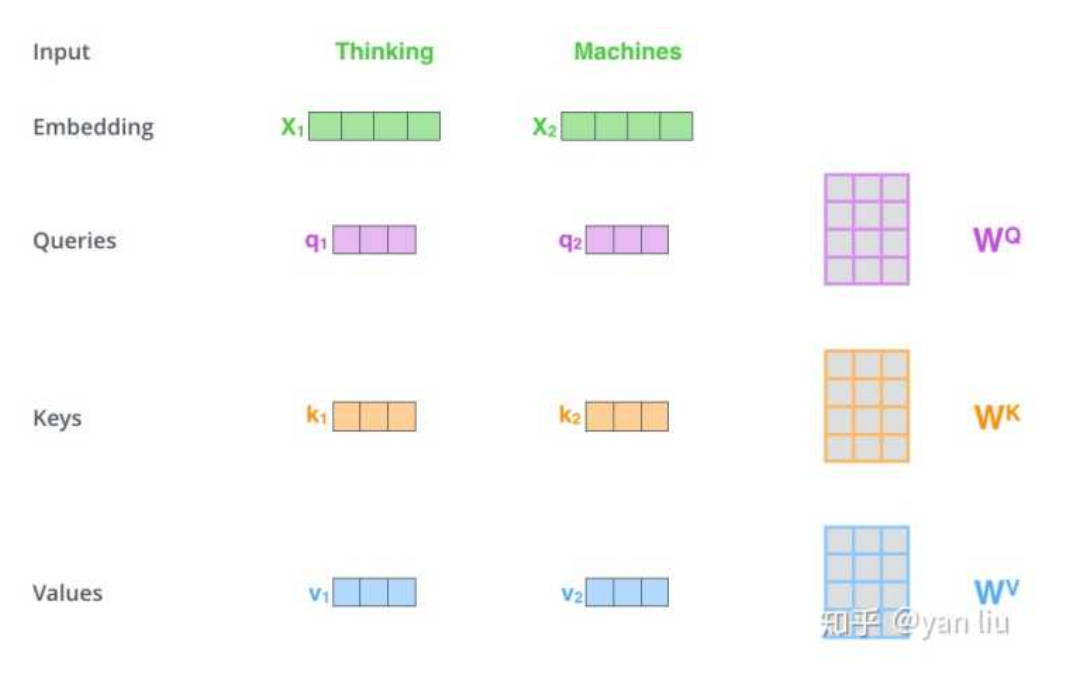

图9：Q，K，V的计算示例图

那么Query，Key，Value是什么意思呢？它们在Attention的计算中扮演着什么角色呢？我们先看一下Attention的计算方法，整个过程可以分成7步：

1. 如上文，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 ： $z=\sum v$ 。

上面步骤的可以表示为图10的形式。

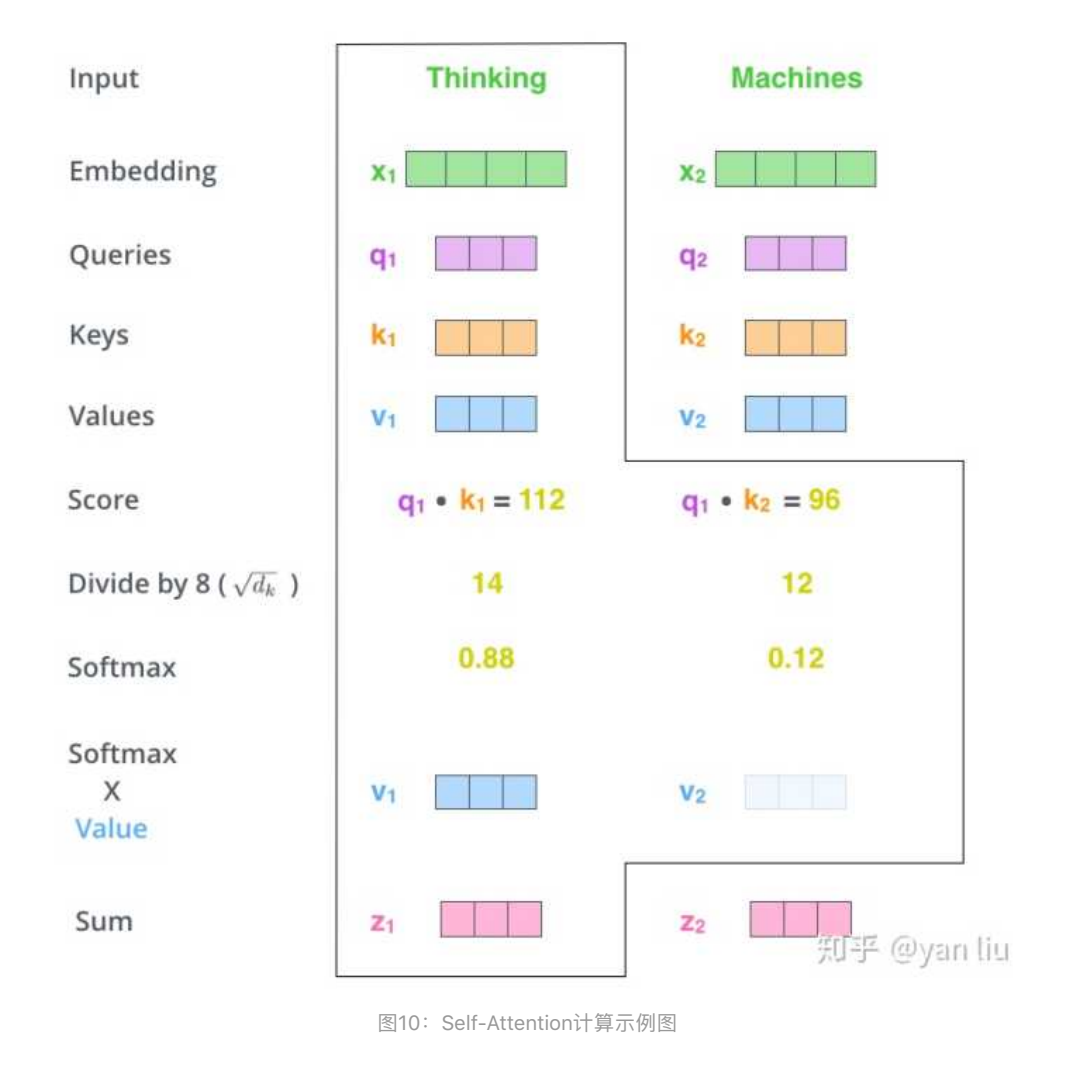

图10：Self-Attention计算示例图

实际计算过程中是采用基于矩阵的计算方式，那么论文中的 Q ， V ， K 的计算方式如图11：

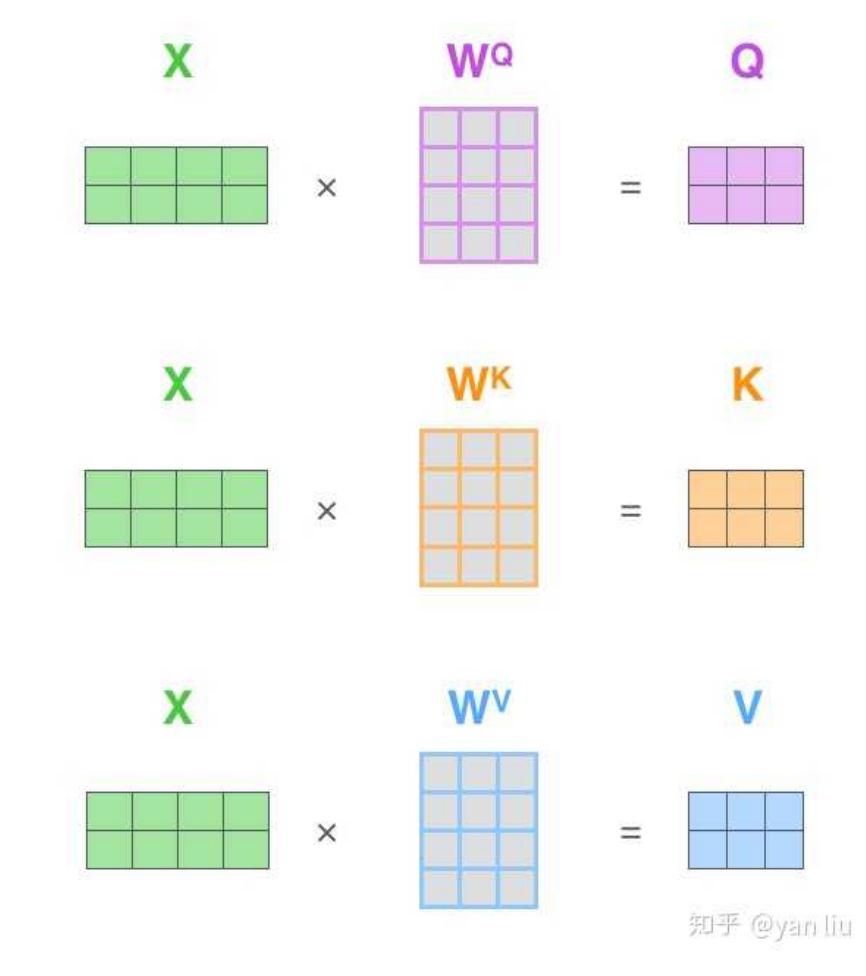

图11：Q，V，K的矩阵表示

图10总结为如图12所示的矩阵形式：

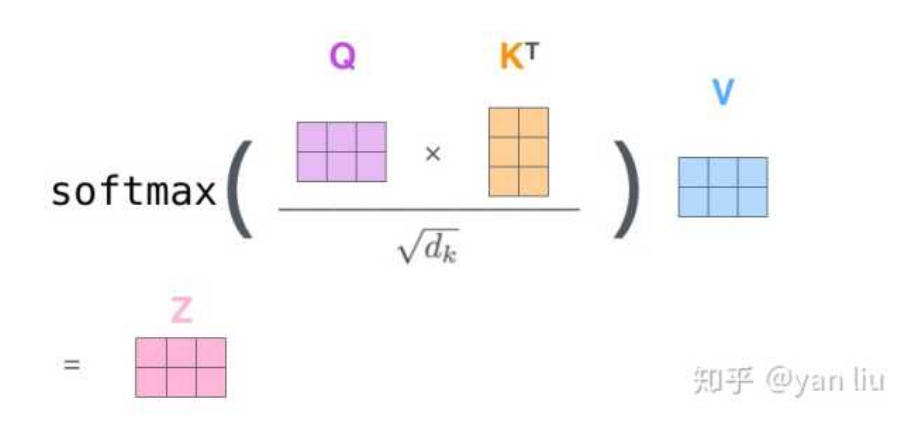

图12：Self-Attention的矩阵表示

这里也就是公式1的计算方式。

在self-attention需要强调的最后一点是其采用了残差网络 [5]中的short-cut结构，目的当然是解决深度学习中的退化问题，得到的最终结果如图13。

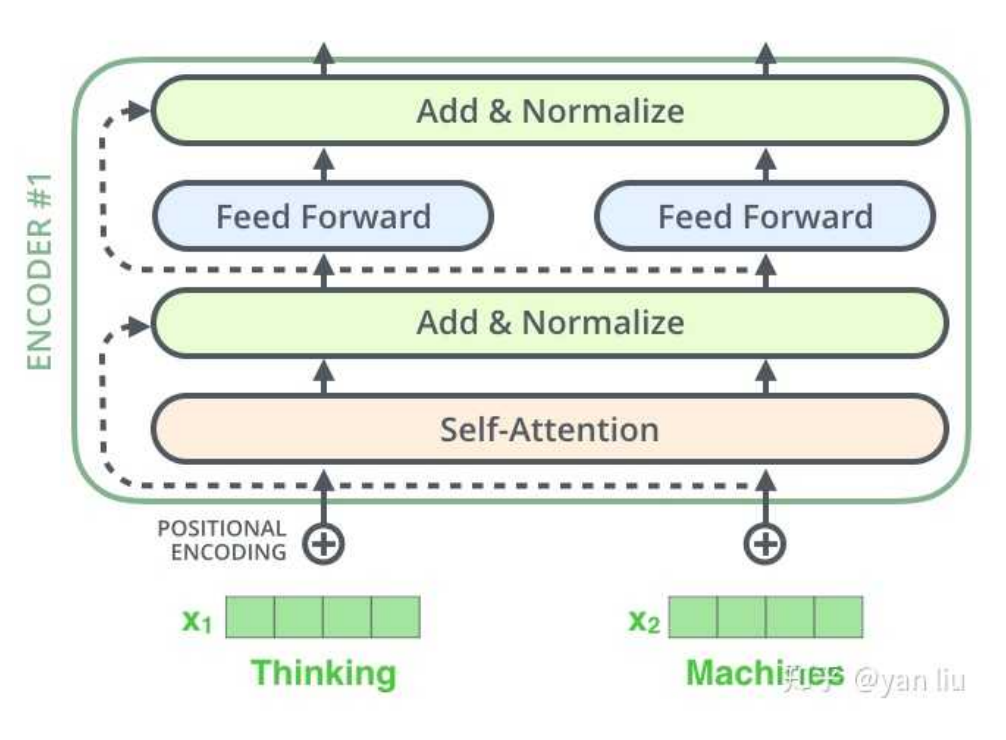

图13：Self-Attention中的short-cut连接

1.3 Multi-Head Attention

Multi-Head Attention相当于 h 个不同的self-attention的集成（ensemble），在这里我们以 $h=8$ 举例说明。Multi-Head Attention的输出分成3步：

1. 将数据 X 分别输入到图13所示的8个self-attention中，得到8个加权后的特征矩阵 $Z_i, i\in\{1,2,\dots,8\}$ 。
2. 将8个 Z_i 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 。

整个过程如图14所示：

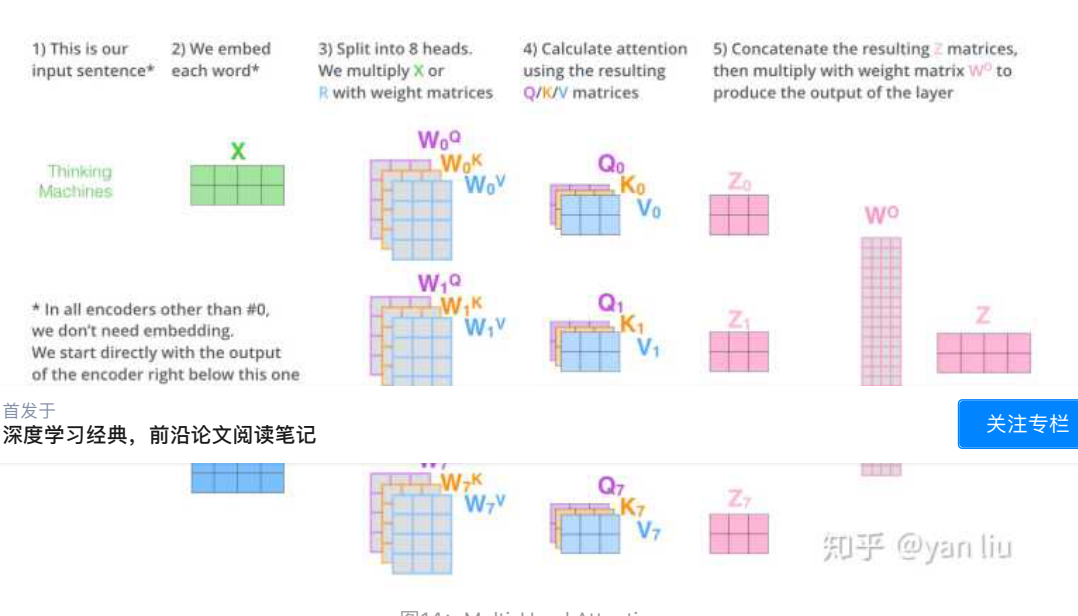

图14：Multi-Head Attention

同self-attention一样，multi-head attention也加入了short-cut机制。

1.4 Encoder-Decoder Attention

在解码器中，Transformer block比编码器中多了个encoder-cecoder attention。在encoder-decoder attention中， Q 来之与解码器的上一个输出， K 和 V 则来自于与编码器的输出。其计算方式完全和图10的过程相同。

由于在机器翻译中，解码过程是一个顺序操作的过程，也就是当解码第 k 个特征向量时，我们只能看到第 $k-1$ 及其之前的解码结果，论文中把这种情况下的multi-head attention叫做masked multi-head attention。

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。此时我们便可以通过CTC等损失函数训练模型了。

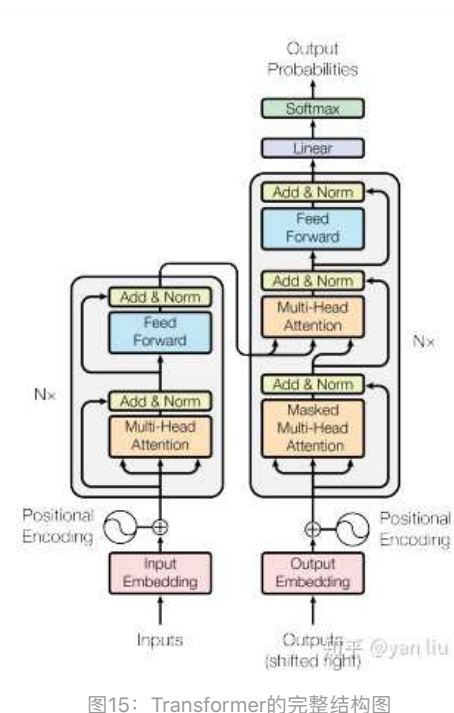

图15：Transformer的完整结构图

2. 位置编码

截止目前为止，我们介绍的Transformer模型并没有捕捉顺序序列的能力，也就是说无论句子的结构怎么打乱，Transformer都会得到类似的结果。换句话说，Transformer只是一个功能更强大的词袋模型而已。

为了解决这个问题，论文中在编码词向量时引入了位置编码（Position Embedding）的特征。具体地说，位置编码会在词向量中加入了单词的位置信息，这样Transformer就能区分不同位置的单词了。

那么怎么编码这个位置信息呢？常见的模式有：a. 根据数据学习；b. 自己设计编码规则。在这里作者采用了第二种方式。那么这个位置编码该是什么样子呢？通常位置编码是一个长度为 d_{model} 的特征向量，这样便于和词向量进行单位加的操作，如图16。

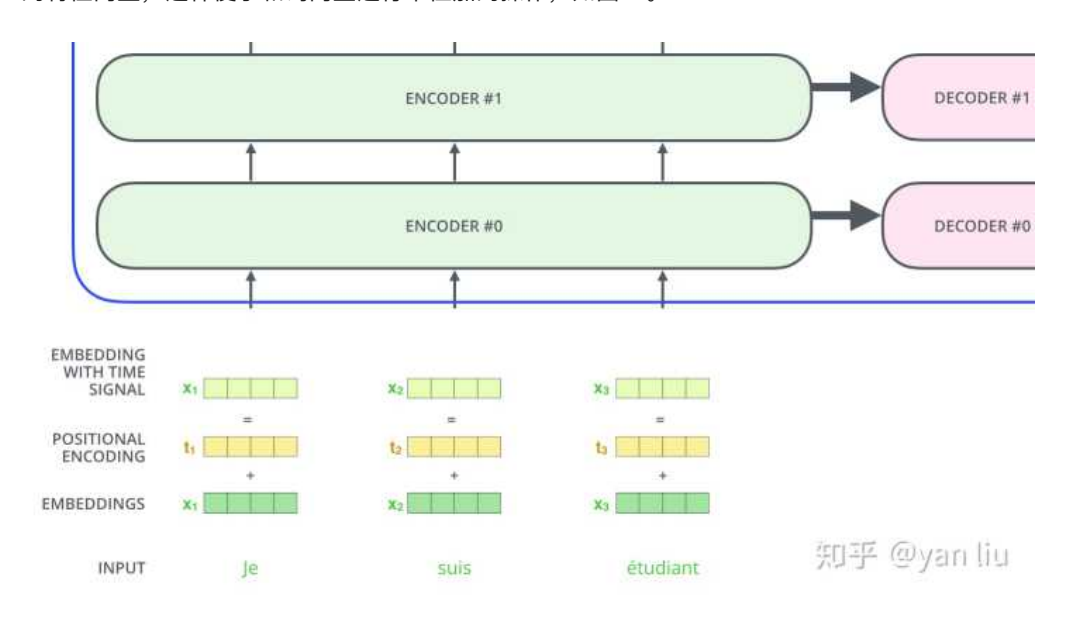

图16：Position Embedding

论文给出的编码公式如下：

$$PE(pos,2i)=sin(\frac{pos}{10000^{\frac{2i}{d_{model}}}}) \quad (3)$$

$$PE(pos,2i+1)=cos(\frac{pos}{10000^{\frac{2i}{d_{model}}}}) \quad (4)$$

在上式中， pos 表示单词的位置， i 表示单词的维度。关于位置编码的实现可在Google开源的算法中 `get_timing_signal_1d()` 函数找到对应的代码。

作者这么设计的原因是考虑到在NLP任务重，除了单词的绝对位置，单词的相对位置也非常重要。根据公式 $sin(\alpha+\beta)=sin\alpha cos\beta+cos\alpha sin\beta$ 以及 $cos(\alpha+\beta)=cos\alpha cos\beta-sin\alpha sin\beta$ ，这表明位置 $k+p$ 的位置向量可以表示为位置 k 的特征向量的线性变化，这为模型捕捉单词之间的相对位置关系提供了非常大的便利。

3. 总结

优点：（1）虽然Transformer最终也没有逃脱传统学习的套路，Transformer也只是一个全连接（或者是一维卷积）加Attention的结合体。但是其设计已经足够有创新，因为其抛弃了在NLP中最根本的RNN或者CNN并且取得了非常不错的效果，算法的设计非常精彩，值得每个深度学习的相关人员仔细研究和品位。（2）Transformer的设计最大的带来性能提升的关键是将任意两个单词的距离是1，这对解决NLP中棘手的长期依赖问题是非常有效的。（3）Transformer不仅仅可以应用在NLP的机器翻译领域，甚至可以不局限于NLP领域，是非常有科研潜力的一个方向。（4）算法的并行性非常好，符合目前的硬件（主要指GPU）环境。

缺点：（1）粗暴的抛弃RNN和CNN虽然非常炫技，但是它也使模型丧失了捕捉局部特征的能力，RNN + CNN + Transformer的结合可能会带来更好的效果。（2）Transformer失去的位置信息其实在NLP中非常重要，而论文中在特征向量中加入Position Embedding也只是一个权宜之计，并没有改变Transformer结构上的固有缺陷。

Reference

[1] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need [C]//Advances in Neural Information Processing Systems. 2017: 5998-6008.

[2] Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate[J]. arXiv preprint arXiv:1409.0473, 2014.

[3] Devlin J, Chang M W, Lee K, et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding[J]. arXiv preprint arXiv:1810.04805, 2018.

[4] jalammar.github.io/illu...

[5] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.

编辑于 2018-11-05

深度学习（Deep Learning） 自然语言处理