NIPS2017｜行为序列建模的方式---Transformer

深度传送门 2020-09-03

以下文章来源于推荐算法的小齿轮 ，作者潜心

推荐算法的小齿轮

记录与分享自己的学习内容，励志做一个推荐算法的小齿轮⚙。

前言

最近想研究序列推荐的内容，刚好看到行为序列建模的BST[1]序列模型运用了Transformer[2]结构，并且美团博客中也提到了“Transformer 在美团搜索排序中的实践”[3]。因此学习了Transformer模型内容，并记录了笔记。本篇文章并没有什么创新，因为基本参考了对Jay Alammar的博客[4]，想要具体了解，可以查看原博客（点击原文链接）。但由于下一篇是想对Transformer中遇到的问题进行汇总与解答（Q&A），所以先将自己整理的内容堆上来，方便参考。

本文约2.7k字，预计阅读15分钟。

Transformer

Transformer，是一个sequence-to-sequence模型，2017年提出。与其他Seq2Seq模型不同的是，它抛弃了传统的RNN与CNN，完全依赖注意机制来构成整个网络的架构，广泛的应用于机器翻译、语音识别等领域，当然也有在序列推荐中有具体的应用。Transformer也是一个encoder-decoder的结构，由自注意力机制（self attention）和前馈神经网络（Feed Forward）堆叠而成。论文中整体的结构如下所示：

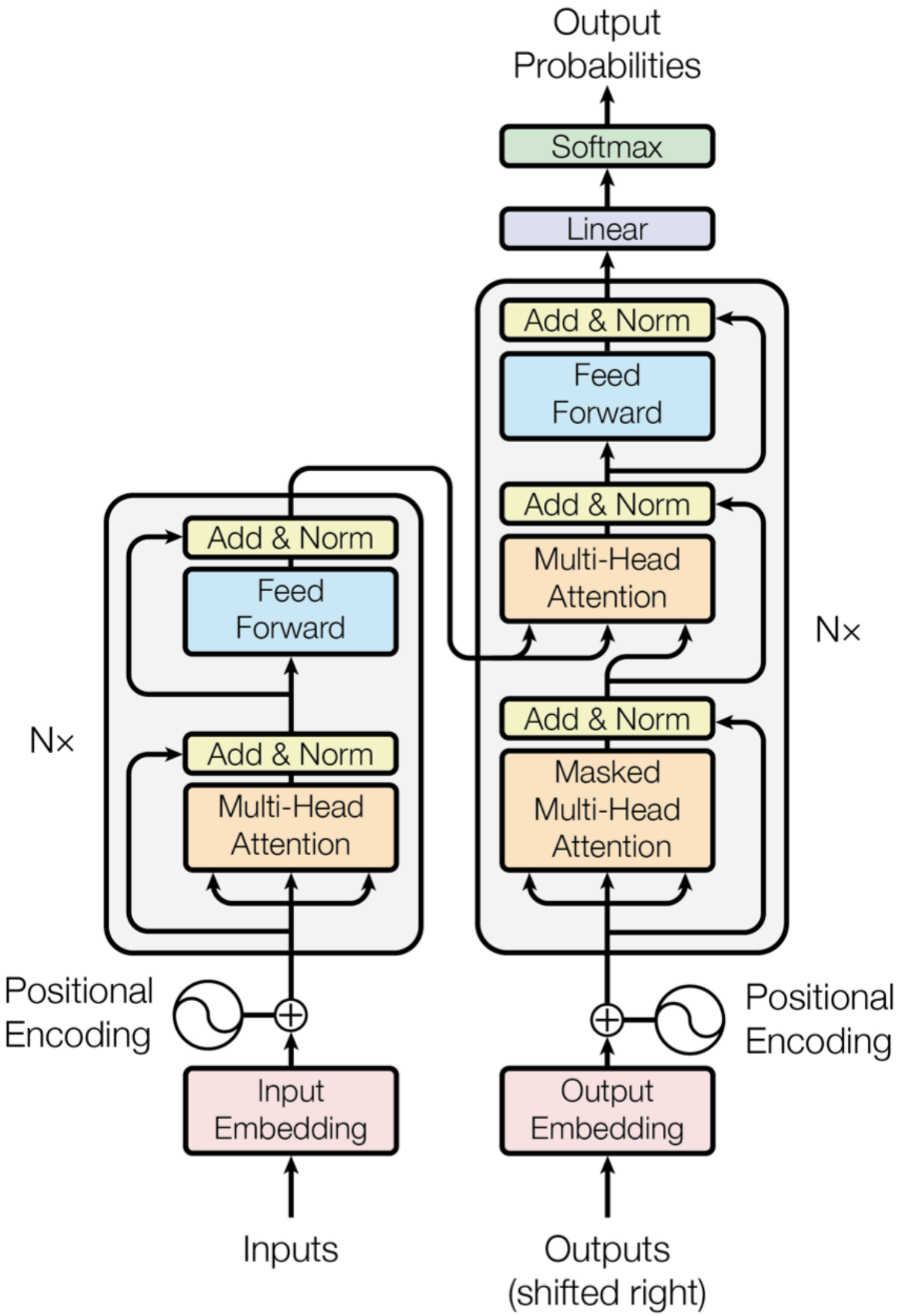


Figure 1: The Transformer - model architecture.

由于注意力机制是整个网络的核心，因此先由它展开。

注意力机制

注意力机制作为一种资源分配方案，将有限的计算资源用来处理更重要的信息，是解决信息超载问题的主要手段。

当神经网络来处理大量的输入信息时，也可以借助注意力机制，选择一些关键的信息输入进行处理，提高神经网络的效率。

用$\mathbf{X}=[x_1,\ldots,x_N]\in\mathbb{R}^{d_k\times N}$表示$N$组输入信息，其中$d_k$维向量$x_n\in\mathbb{R}^{d_k},n\in[1,N]$表示一组输入的信息（向量）。注意力机制的计算可以分为两步：

1. 在所有输入信息上计算**「注意力分布」**；
2. 根据注意力分布来计算输入信息的加权平均；

「注意力分布：」

为了从N个输入向量$[x_1,\ldots,x_n]$中选择出和某个特定任务相关的信息，需要引入一个和任务相关的表示，即**「查询向量$q\in\mathbb{R}^{d_k}$」**，通过一个打分函数来计算**「每个输入向量和查询向量之间的相关性」**。给定一个和任务相关的查询量量$\mathbf{q}$，用注意力变量$z\in[1,N]$来表示被选择信息的索引位置，即$z=n$表示选择了第n个输入向量。首先计算在给定$\mathbf{q}$和$\mathbf{X}$下，选择第i个输入向量的概率α_n，

$$\begin{aligned}\alpha_n &= p(z=n|\mathbf{X},\mathbf{q})\\ &= softmax(s(\mathbf{x}_n,\mathbf{q}))\\ &= \frac{exp(s(\mathbf{x}_n,\mathbf{q}))}{\sum_{j=1}^{N}exp(s(\mathbf{x}_j,\mathbf{q}))}\end{aligned}$$

α_n称为**「注意力分布」**，也可以说是在给定任务相关的查询q时，第n个输入向量受关注的程度。$s(\mathbf{x},\mathbf{q})$为注意力打分函数，主要包括：

1. 加性模型：$s(\mathbf{x},\mathbf{q})=\mathbf{v}^T tanh(\mathbf{W}\mathbf{x}+\mathbf{U}\mathbf{q})$
2. 点积模型：$s(\mathbf{x},\mathbf{q})=\mathbf{x^T q}$
3. 缩放点积模型：$s(\mathbf{x},\mathbf{q})=\frac{\mathbf{x^T q}}{\sqrt{d_k}}$
4. 双线性模型：$s(\mathbf{x},\mathbf{q})=\mathbf{x^T W q}$

其中$\mathbf{W},\mathbf{U},\mathbf{v}$为可学习的参数，$D$为输入向量的维度。

在Transformer中，注意力打分函数选择**「缩放点积模型」**。文章先解释了使用点击模型的**「原因」**：

> Additive attention computes the compatibility function using a feed-forward network with a single hidden layer. While the two are similar in theoretical complexity, dot-product attention is much faster and more space-efficient in practice, since it can be implemented using highly optimized matrix multiplication code.

简单来说，就是点积模型可以使用矩阵乘法进行计算（GPU）。

然后在点积的基础上加入缩放是因为：当输入维度d_k较高时，点击模型的值通常有较大的方差，从而导致Softmax函数的梯度比较小，而缩放点击模型可以很好的解决这个问题。

> We suspect that for large values of d_k , the dot products grow large in magnitude, pushing the softmax function into regions where it has extremely small gradients. To counteract this effect, we scale the dot products by $\frac{1}{\sqrt{d_k}}$.

「加权平均：」

使用加权平均对所有的输入信息进行汇总：

$$att(\mathbf{X}, \mathbf{q}) = \sum_{n=1}^{N} \alpha_n x_n$$

以上便称为注意力机制。

Query-Key-Value

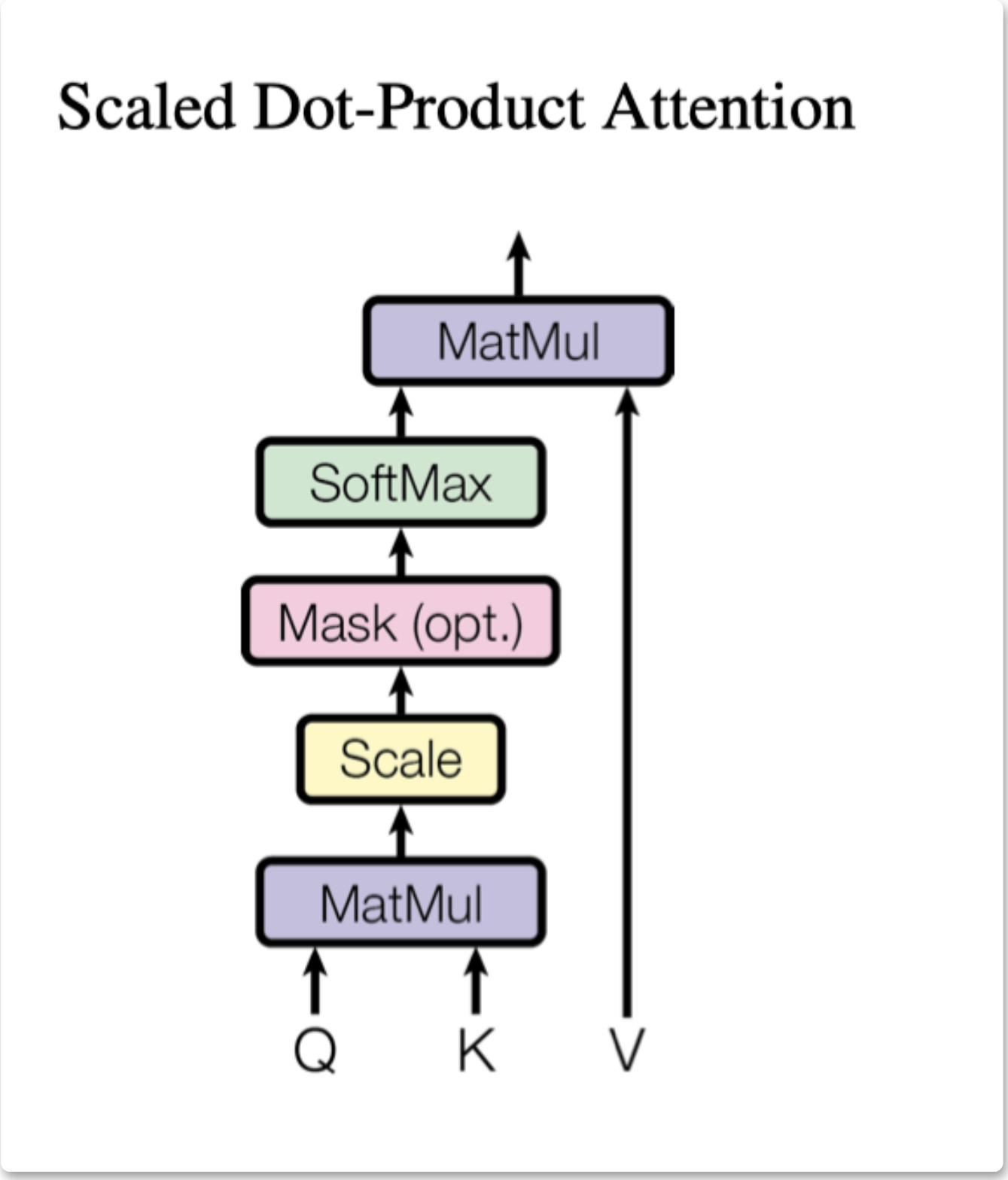
以上是比较直观的注意力机制的解释，但在大部分论文中，都会使用key-value的格式来表示**「输入信息」**，即计算注意力分布时使用键key，而值value则用来计算聚合的信息，因此上述内容就是key=value。

注意力函数为：

$$\text{Attention}\left(q, K, V\right) = \text{softmax}\left(\frac{qK^T}{\sqrt{d_k}}\right)V$$

如下图所示（图来源于博客）：

论文中具体结构如下所示：

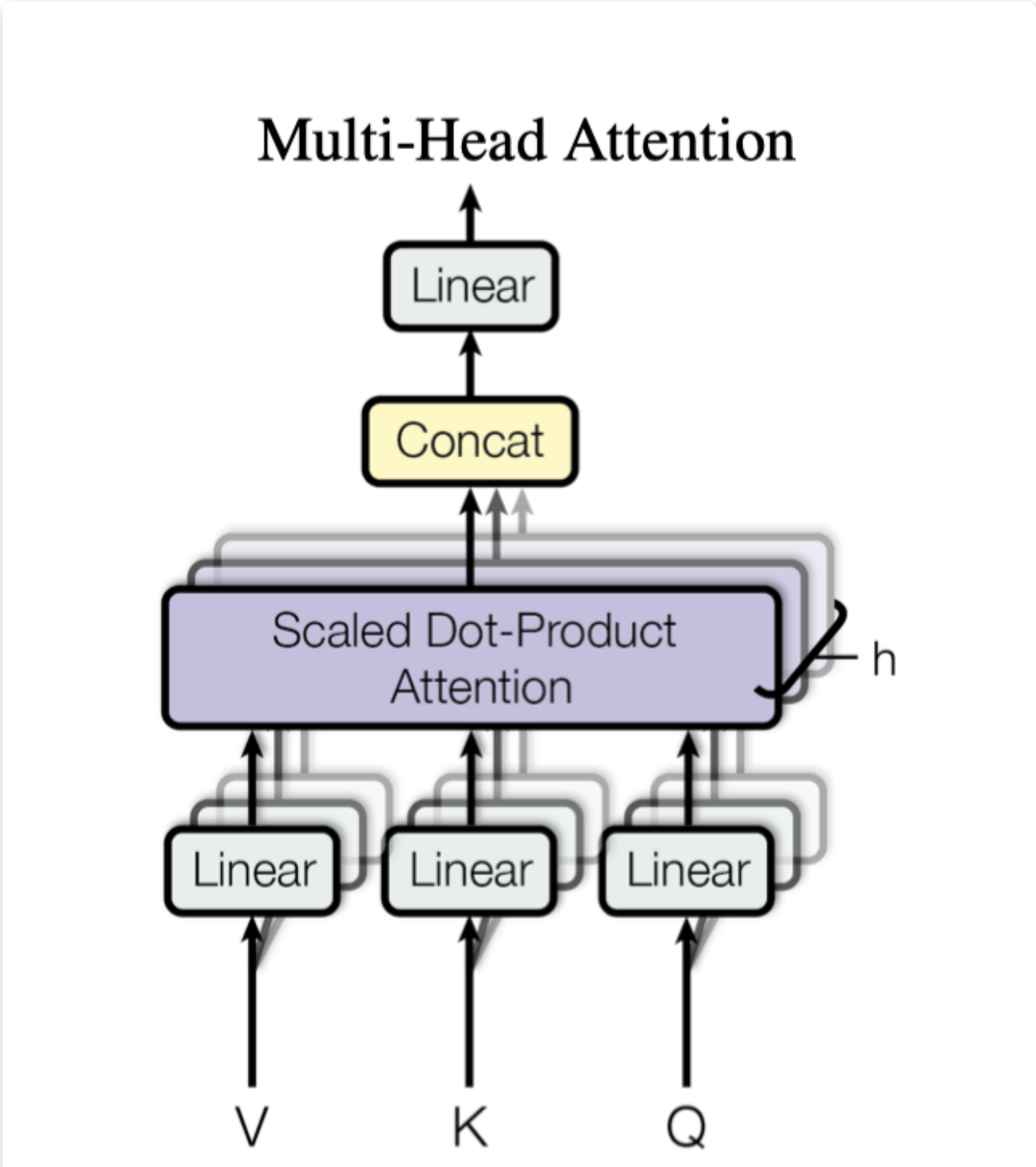


多头注意力机制

多头注意力（Multi-Head Attention）就是将查询向量q扩展为多个查询$Q = [q_1, q_2, \ldots, q_h]$来并行地从输入中选取多组信息，每个注意力关注输入信息的不同部分：

$$\begin{aligned}\text{MultiHead}(Q, K, V) &= \text{Concat}\left(\text{head}_1, \ldots, \text{head}_h\right)W^O \\ \text{where head}_i &= \text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right)\end{aligned}$$

其中，$W_i^Q \in \mathbb{R}^{N\times d_k}, W_i^K \in \mathbb{R}^{N\times d_k}, W_i^V \in \mathbb{R}^{hd_k\times d_{model}}$，为学习的参数矩阵，$d_{model}$为最后所需的维度。



自注意力机制模型

「引入自注意力机制的原因：」

> 神经网络处理一个变长的向量序列时，通常可以使用CNN或RNN编码得到一个相同长度的向量输出序列。但CNN与RNN都是一种局部编码方式，只建模了输入信息的局部依赖关系。RNN由于信息传递的容量以及梯度消失问题，实际也只能建立短距离依赖关系。
>
> 建立输入序列之间的长距离依赖关系，可以使用两种方法：1、增加网络层数；2、使用全连接网络；
>
> 但全连接网络无法处理变长的输入序列。因此可以引入注意力机制来“动态”地生成不同连接的权重，这就是自注意力机制。---《神经网络与深度学习》

「自注意力机制是整个Transformer的核心」。具体步骤如下：

1. 定义输入序列为$X=[x_1,\cdots,x_N]\in\mathbb{R}^{D_x\times N}$，输出序列为$H=[h_1,\ldots,h_N]\in\mathbb{R}^{D_v\times N}$；

2. 对于整个输入序列X，生成三个向量序列：

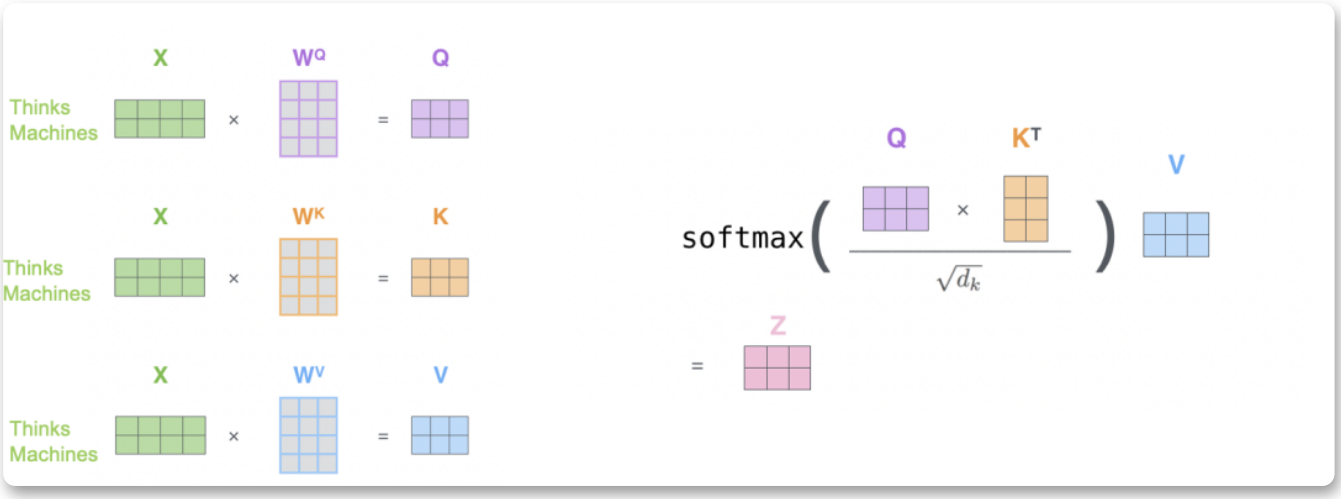
$$Q=W_QX\in\mathbb{R}^{D_k\times N}$$
$$K=W_KX\in\mathbb{R}^{D_k\times N}$$
$$V=W_VX\in\mathbb{R}^{D_v\times N}$$

   其中$W_Q\in\mathbb{R}^{D_k\times D_x},W_K\in\mathbb{R}^{D_k\times D_x},W_V\in\mathbb{R}^{D_v\times D_x}$，且$Q=[q_1,\ldots,q_N],K=[k_1,\ldots,k_N],V=[v1,\ldots,v_N]$，分别为查询向量、键向量、值向量构成的矩阵（**「通过输入序列产生」**）。

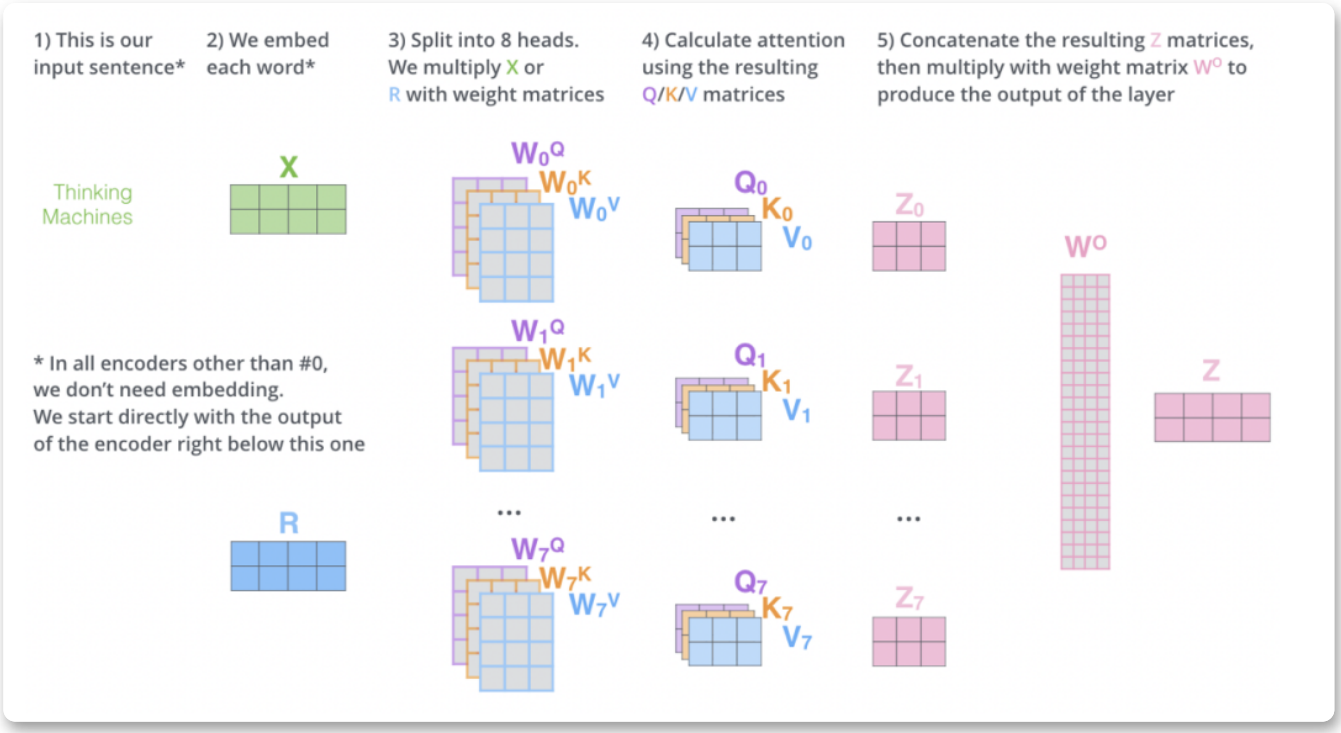
3. 使用缩放点积作为注意力打分函数，那么输出向量序列为：

$$H=softmax(\frac{K^TQ}{\sqrt{D_k}})V$$

$N=2,D_x=3,D_v=3$，自注意力机制如下所示：



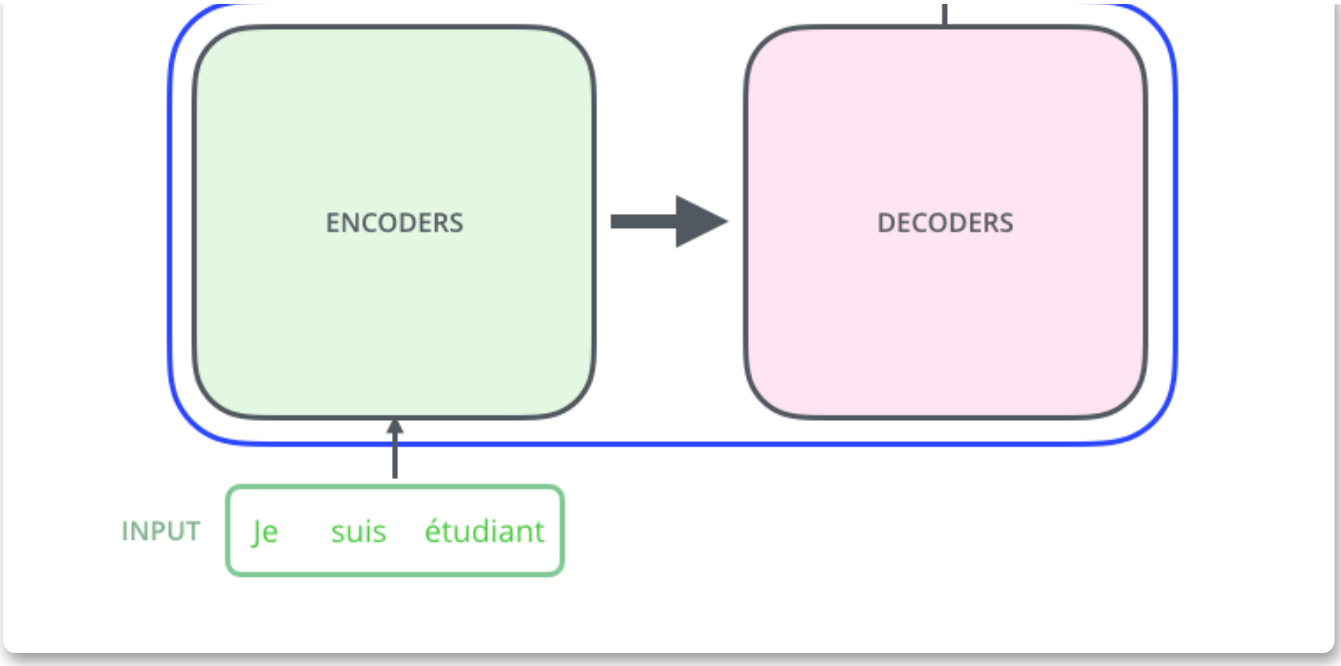
论文中，Transformer内部具体使用的是一个**「多头自注意力机制」**，即多头注意力机制的概念+自注意力机制：



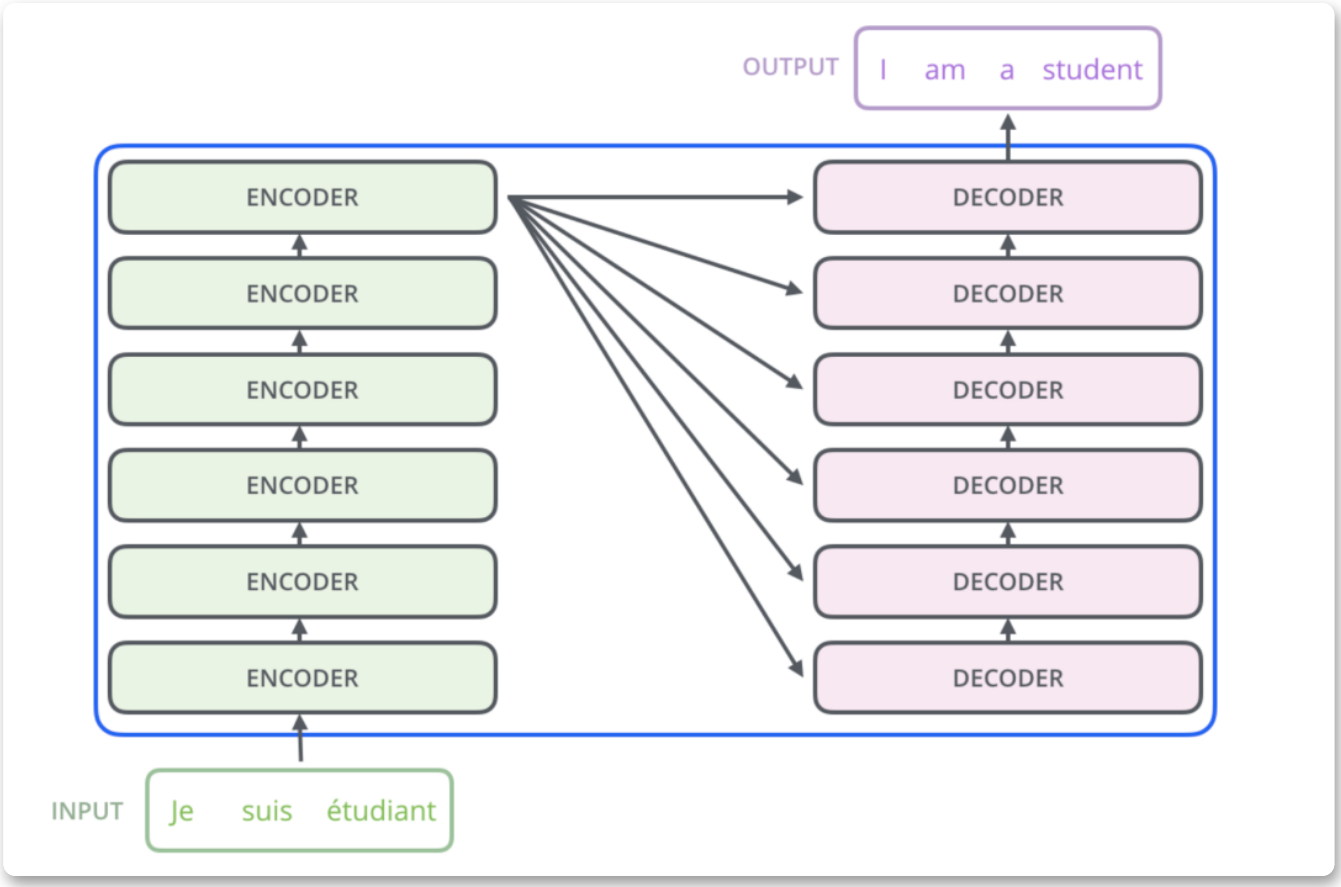
以上便是注意力机制的所有内容。

模型结构

Transformer是一个Encoder-Decoder结构。



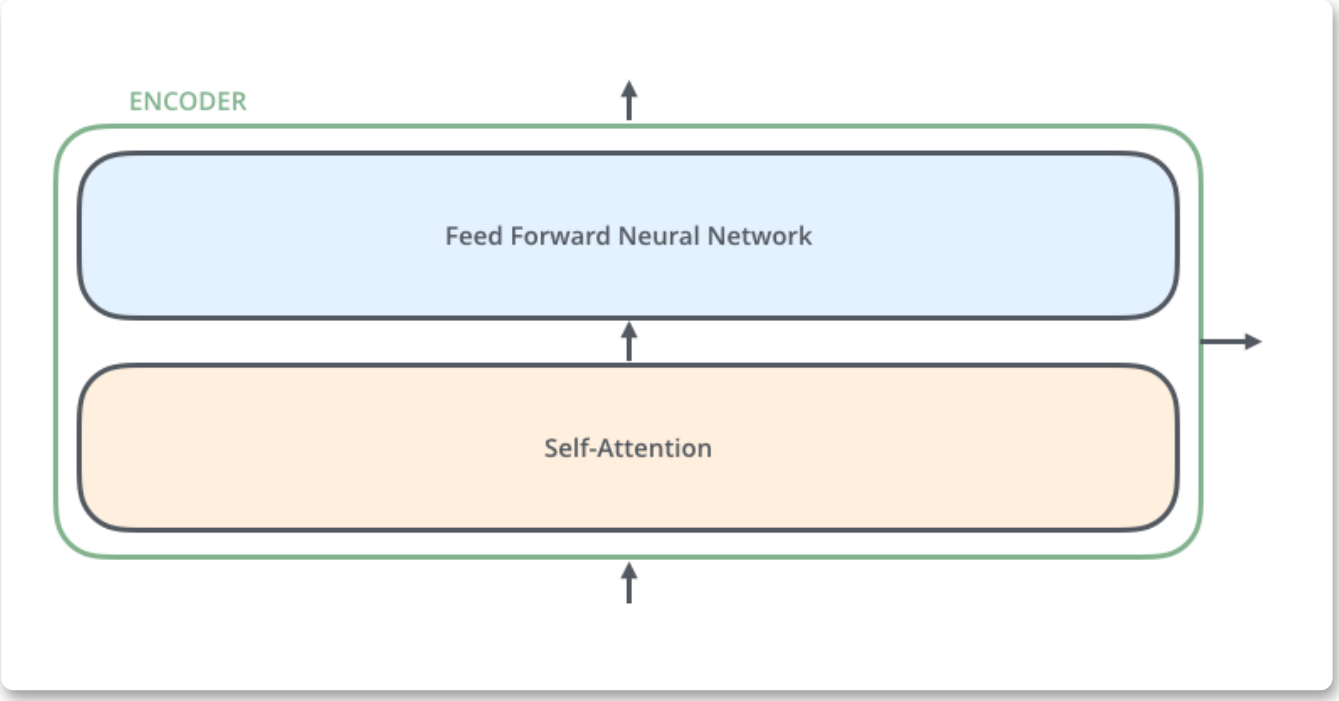
论文中提到，Transformer的编码层由6个相同的层（dentical layers）堆叠而成，解码层同样由6个相同的层堆叠而成，如下所示：



编码层

「总体结构：」

编码层由六个相同的层构成，而每一个又则由两个子层构成：第一个便是上述提到的**「多头自注意力机制层」**，第二个便是简单的**「全连接的前向网络」**，如下所示：



「全连接网络：」

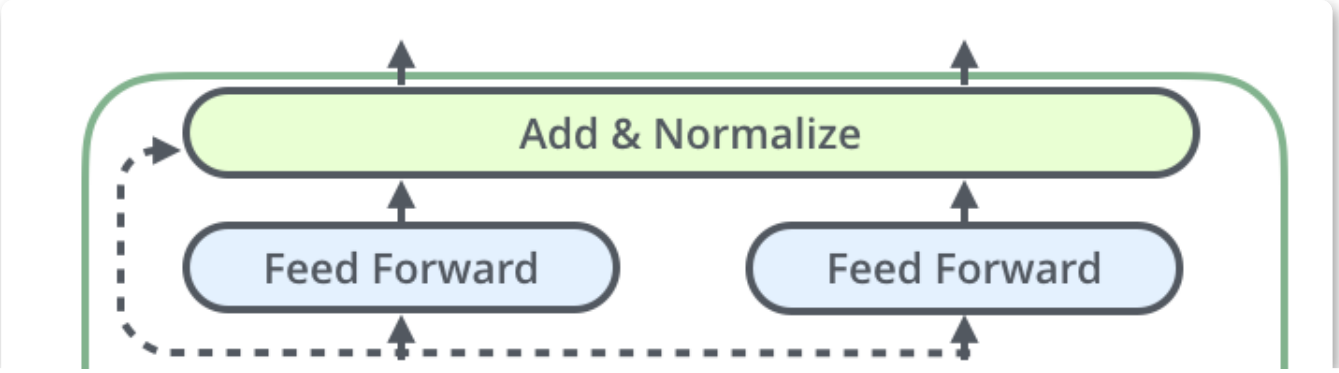
多头自注意力机制在Attention部分已经提到，而全连接网络部分比较简单，即有两个带有Relu激活函数的线性转换构成：

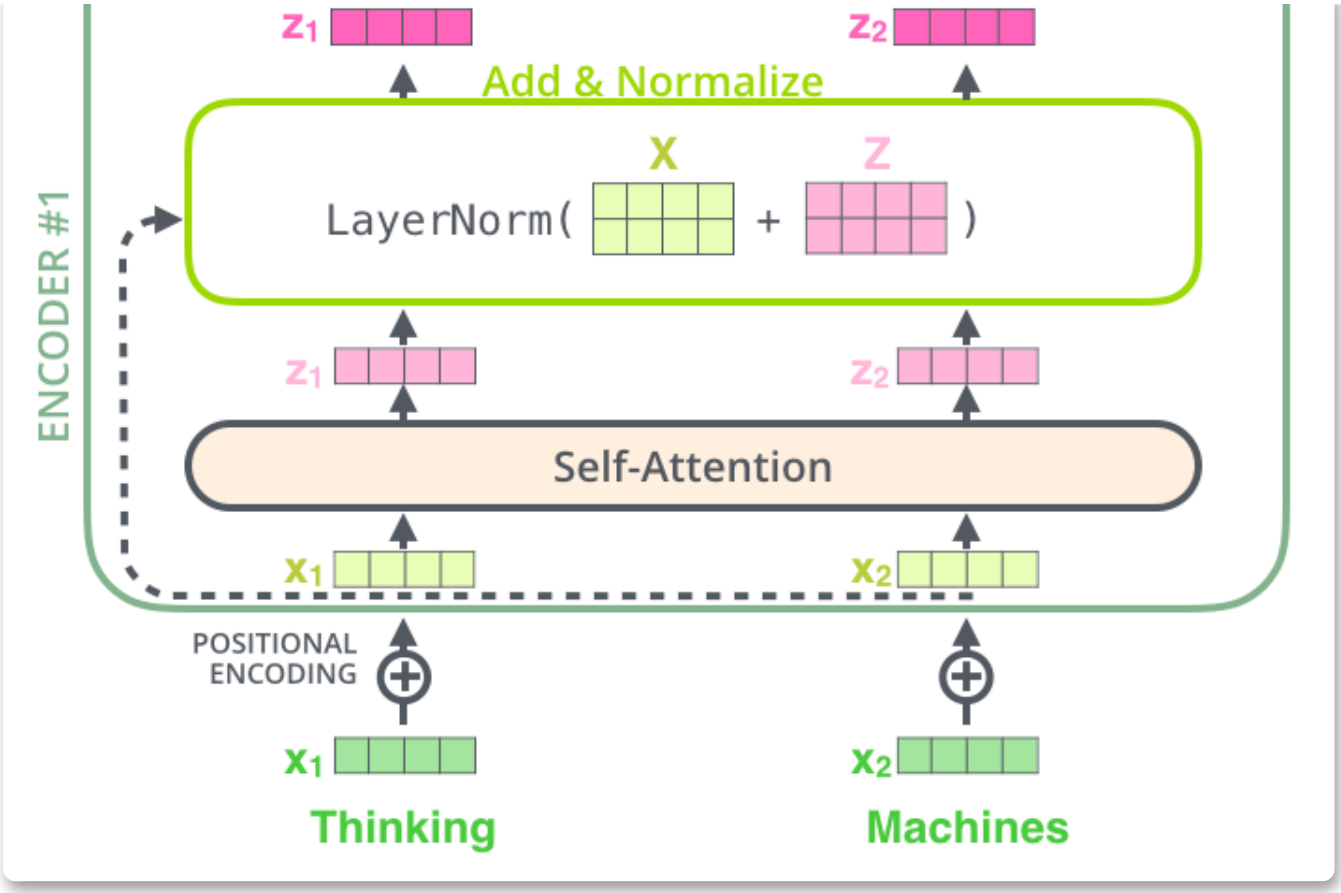
$$FNN(x) = max(0, xW_1 + b_1)W_2 + b_2$$

其中x为自注意力层的输出。

「残差连接：」

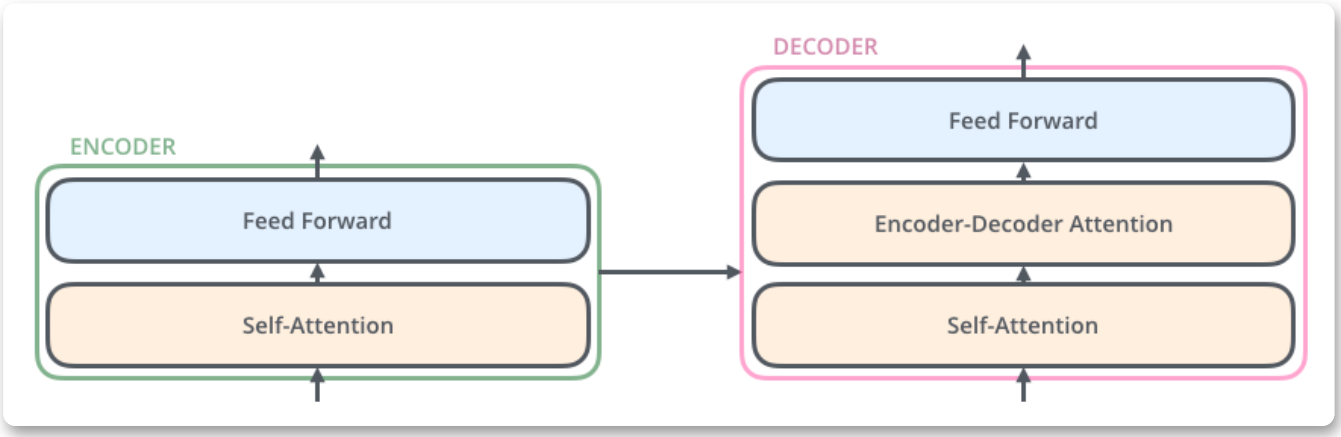
观察论文给出的整体模型，我们发现在每一层中，还引入了一个**「残差连接」**（residual connection），之后通过了一个**「层的Normalization」**。最终编码层每一层的结构如下：



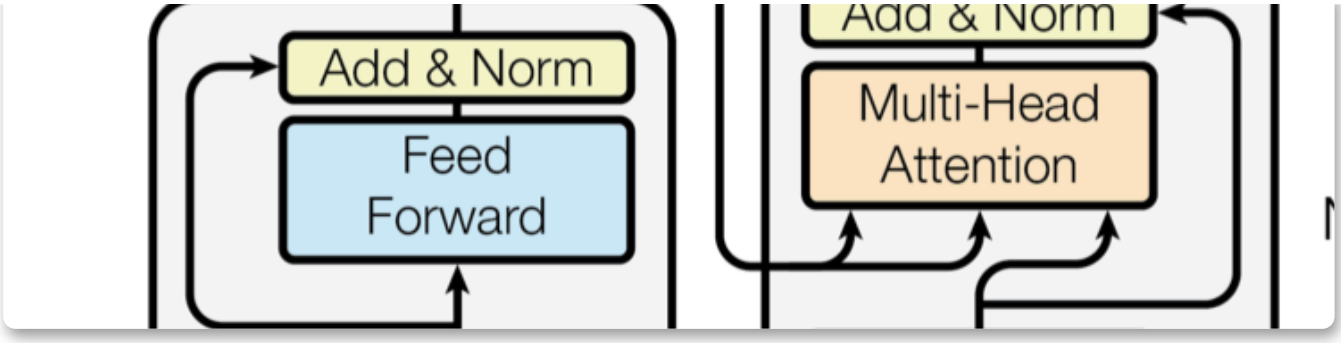


解码层

解码层也是与编码层一样，具有相同的6层。但每一层的结构却与编码层不一样，是由三个子层所构成：**「多头自注意力机制层、Encoder-Decoder的注意力层和全联接的前向网络层」**。相比于编码层，Encoder-Decoder的注意力层主要是为了关注输入的相关部分。



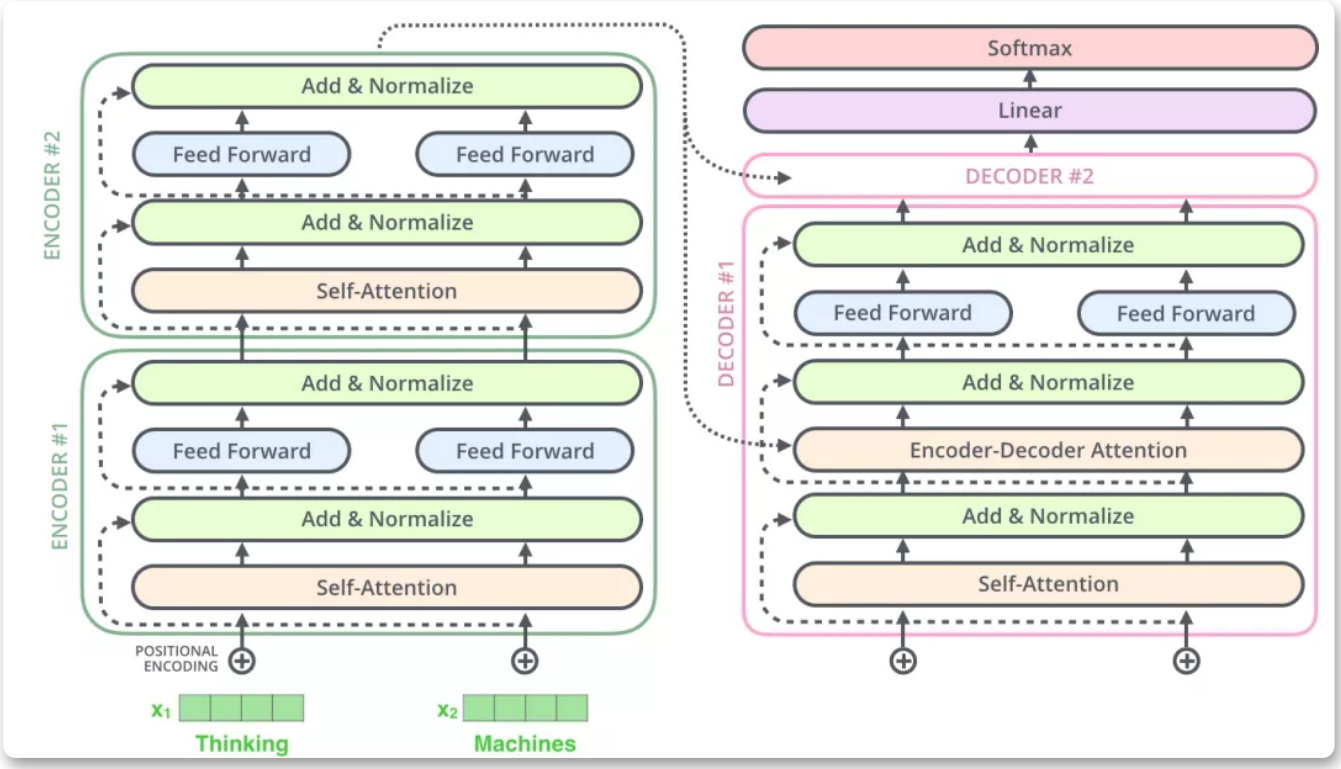
在解码层，我们重点应该关注的是 `Encoder-Decoder Attention` 。



通过模型结构图，**「发现编码层最后的输出作为了中间层的两个输入（Key-Value）」**，而在**「第一个子层多头自注意力机制的输出作为Query」**。该部分就是编码层与解码层的**「本质区别」**。

Encoder-Decoder

因此两者的总体结构为：



Positional Encoding

自注意力机制的权重计算都是依赖于Q与K的相关性的，并没有考虑到**「输入的位置信息」**。即输入序列不管怎么打乱，那么最后Transformer都是得到类似的结果。

为了解决上述问题，Transformer在输入与编码层之间引入**「位置编码」**进行修正。

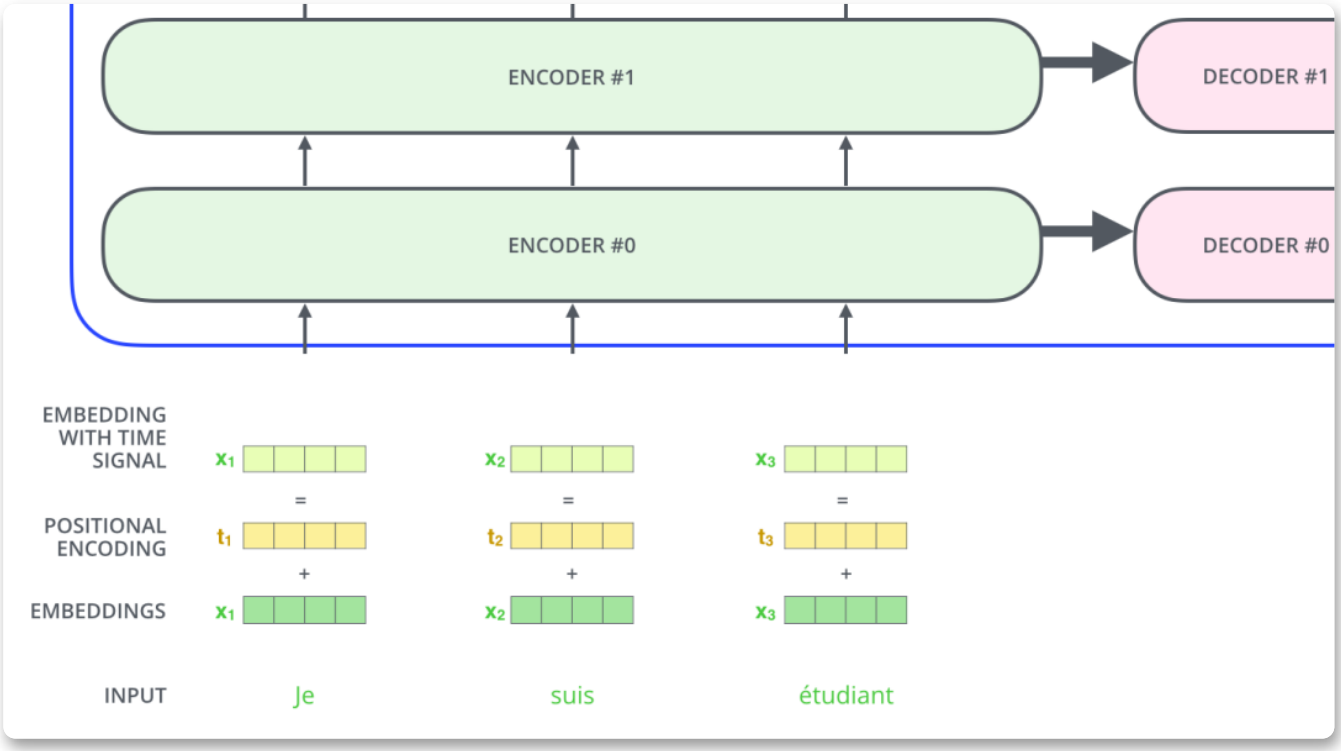
对于一个输入序列$x_1 \in \mathbb{R}^{D \times T}$，经过embedding、位置编码，最后输出为：

$$H^{(0)} = [e_{x_1} + p_1, \ldots, e_{x_T} + p_T]$$

其中e_{x_1}表示embedding的结构，$p_{pos} \in \mathbb{R}^D$为位置pos的向量表示，即位置编码。其中p_{pos}可以作为可学习的参数，也可以通过预定义的方式，论文中定义如下：

$$PE_{(pos,2i)} = \sin\left(pos/10000^{2i/D}\right)$$
$$PE_{(pos,2i+1)} = \cos\left(pos/10000^{2i/D}\right)$$

其中$PE_{(pos,2i)}$表示第pos位置的编码向量的第$2i$维，D表示编码向量的维度。



具体参数

论文中指定的参数如下：

- 为了**「方便残差连接」**，Embedding输出、每个子层的输出维度都为：$d_{model} = 512$;
- Query与Key的维度：$d_k = 64$;
- 多头注意力机制，Query的个数为：$h = 8$;
- Value的维度：$d_v = 8$;
- 因此多头自注意力机制的输出维度为：$d_{model} = hd_v = 512$

总结

Transformer模型是一个值得研究的Seq2Seq模型，BERT核心内容也是Transformer。接下来会对Transformer中一些常见的疑问进行梳理，以Q&A的形式记录。

【题外话】：学习Transformer最好的方式是阅读原文[3]+博客[4]。

参考资料

[1] Chen Q, Zhao H, Li W, et al. Behavior sequence transformer for e-commerce recommendation in Alibaba[C]//Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data. 2019: 1-4.
[2] https://tech.meituan.com/2020/04/16/transformer-in-meituan.html
[3] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[C]//Advances in neural information processing systems. 2017: 5998-6008.
[4] https://jalammar.github.io/illustrated-transformer/?spm=ata.13261165.0.0.34fb48aaFxc8Jt
[5] 神经网络与深度学习

关于深度传送门

深度传送门是一个专注于深度推荐系统与CTR预估的交流社区，传送推荐、广告以及NLP等相关领域工业界第一手的论文、资源等相关技术分享，欢迎关注！加技术交流群请添加小助手deepdeliver，备注姓名+学校/公司+方向。

你点的每个“在看”，我都认真当成了喜欢

阅读原文