# Transformer: Attention的集大成者

原创 张雨石 雨石记 7月25日

收录于话题
#Bert那些事儿

24个

最近要开始使用Transformer去做一些事情了，特地把与此相关的知识点记录下来，构建相关的、完整的知识结构体系，

以下是要写的文章，本文是这个系列的第一篇：

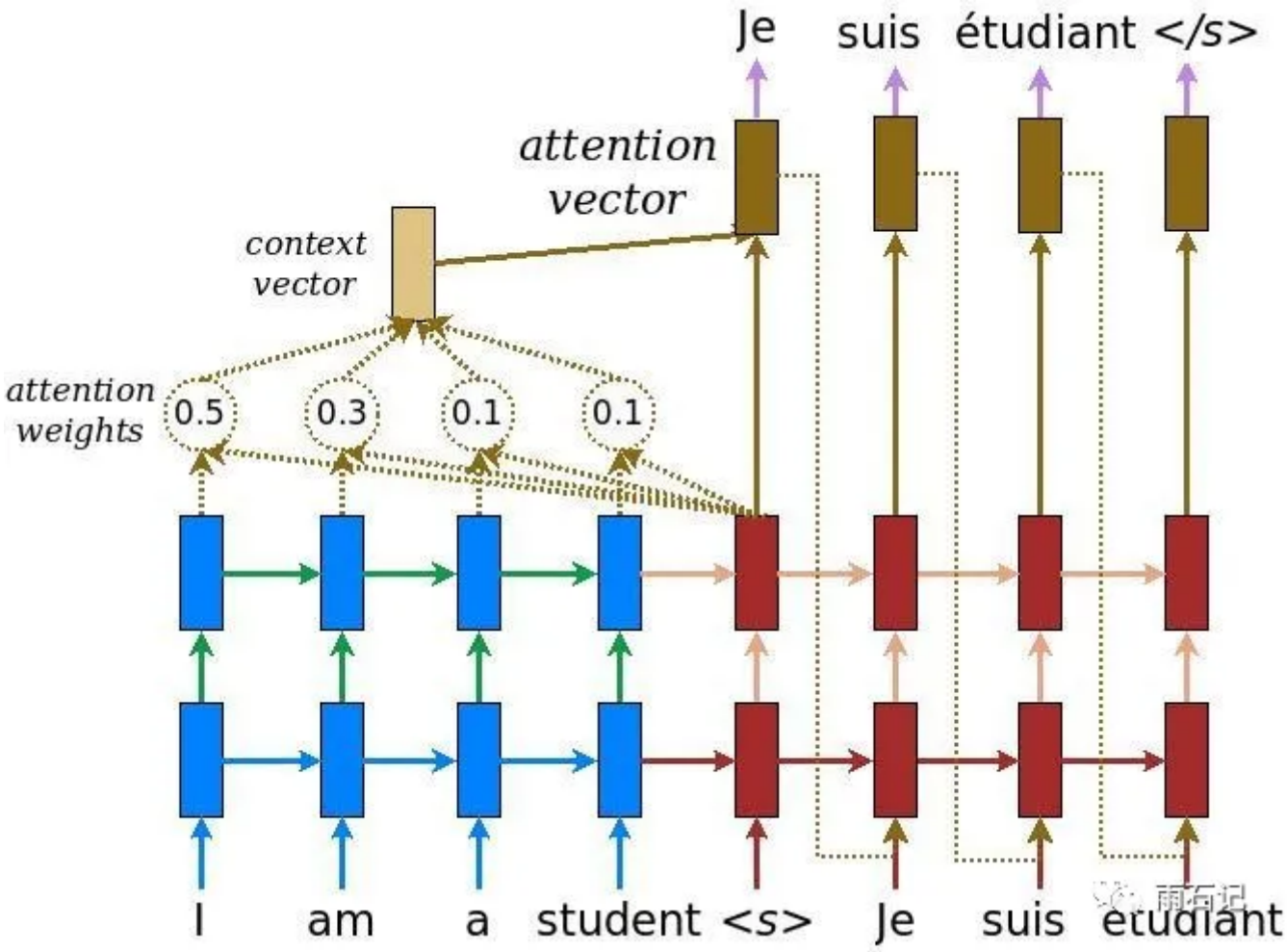
- Transformer:Attention集大成者
- GPT (待续)
- Bert (待续)
- Bert与模型压缩 (待续)
  - ALBert
  - MobileBert
  - 更多
- Bert与AutoML (待续)
- 线性Transformer (待续)
- Bert变种 (待续)
  - Roberta
  - Reformer
  - Longformer
  - T5
  - 更多
- 更多待续

## Attention

Transformer模型是机器翻译领域的一个全部基于attention的模型。那么什么是attention呢？且看下图。图中是一个seq2seq+attention的机器翻译模型。机器翻译问题中分为源语言和目标语言，图中是英语到西班牙语的翻译，英语是源语言，西班牙语是目标语言。

在图中，如果不看左上角的attention weights，那么可以看做是两个循环神经网络的拼接，第一个循环神经网络接受源语言的输入，第二个循环神经网络一边接受目标语言中的词语，一边输出目标语言的下一个词。也就相当于是模型利用源语言中的所有信息和目标语言中的已知信息来预测目标语言中的下一个词，直到预测完为止。

这样的坏处就是如果句子太长，那么循环神经网络中保存的离当前位置比较远的信息就可能找不到，从而导致效果差。



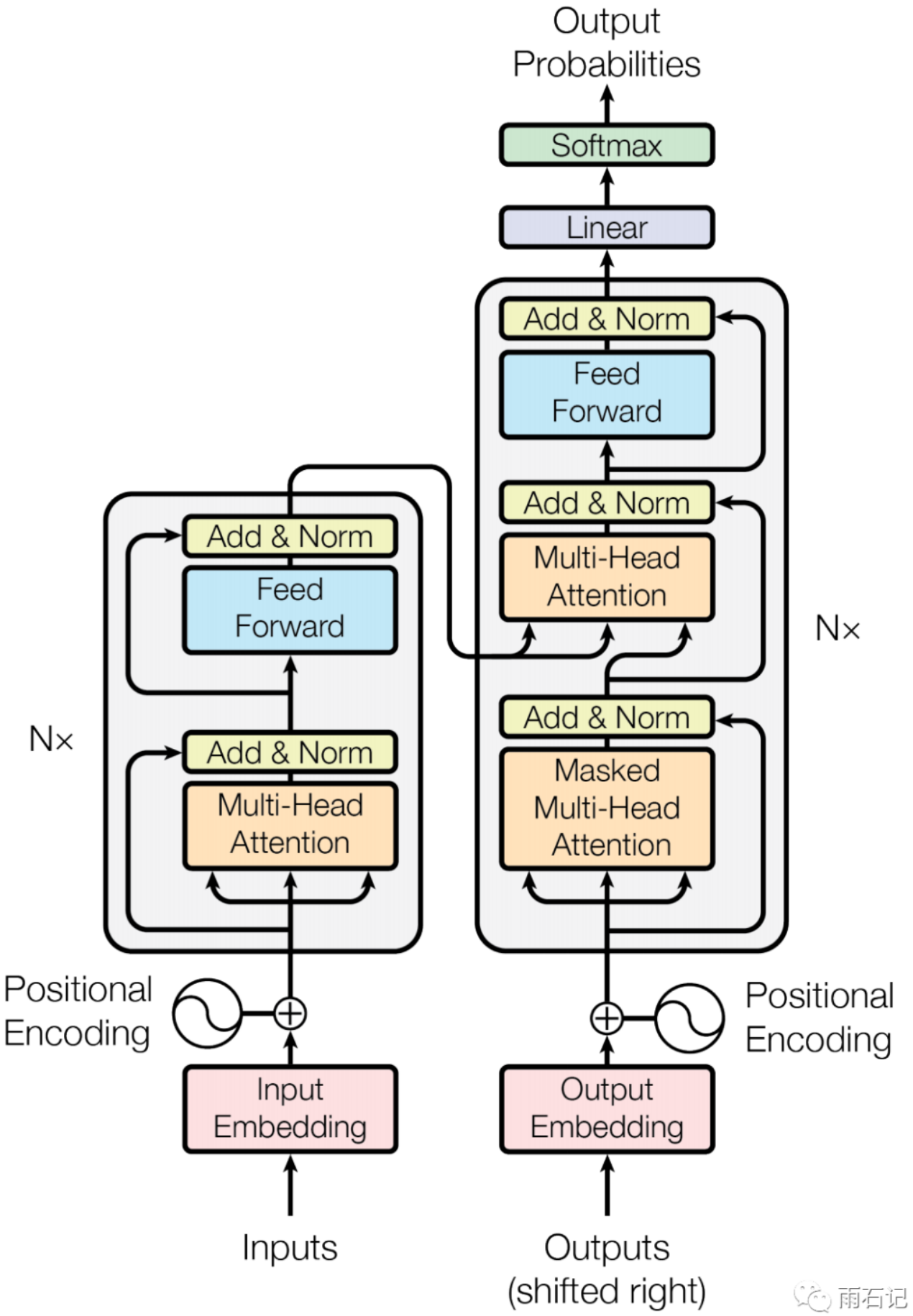

所以左上角的attention应运而生，这个的目的就是在目标语言预测每个词的时候，都会去让当前步的状态去和之前的源语言中的每一步的状态去做Attention。

说了那么久，解释一下attention，attention说白了就是权重计算和加权求和。图上的循环神经网络中的每一步都会输出一个向量，在预测目标语言到某一步时，用当前步的向量去和源语言中的每一步的向量去做内积，然后经过softmax得到归一化后的权重，再用权重去把源语言上的每一步的向量去做加权平均。然后做预测的时候也作为输入进入全连接层。

## Transformer模型

而Transformer模型则是在seq2seq+Attention模型的基础上把循环神经网络的部分给去掉了，取而代之的也是attention结构。

先看一下总图，如下，别害怕，我们慢慢拆分。



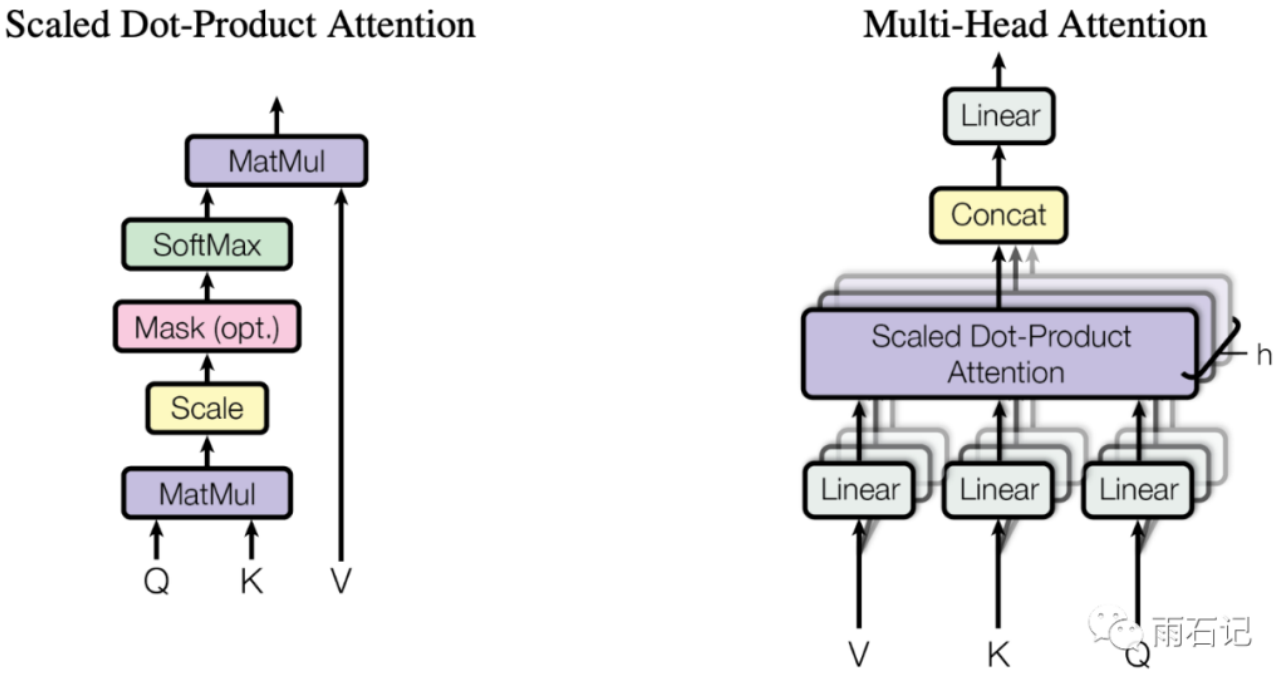
跟前面一样，这个模型还是两部分，左边处理源语言，称之为Encoder，右边处理目标语言，被称为Decoder，分别由N个Block组成。然后每个block都有这么几个模块：

- Multi-Head Attention
- Masked Multi-Head Attention
- Add & Norm
- Feed Forward
- Positional Encoding
- Linear

其中， Feed Forward和Linear是神经网络的基本操作全连接层，Add & Norm以及延伸出来的一条侧边也是一个常见的神经网络结构残差连接，所以这三个我们就不细讲了。

## Multi-Head Attention

下图中展示了Multi-Head Attention的计算原理，Multi-Head Attention是由多个Scaled Dot-Product Attention的函数组合而成的。




Scaled Dot-Product Attention的计算公式如下：

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

公式还是我们上面提到的计算权重和加权求和两个老套路。所区别的是，算权重是Q和K去算，算加权和是QK的结果和V去算。

在这里提两个问题：

> 问题1：Q,K,V是怎么来的？
>
> 答：输入的每个词经过Input Embedding后会变成一个向量，然后这个向量会分别经过一个矩阵变换得到Q,K,V。

> 问题2：Q,K,V分别的含义是什么，
>
> 答：Q代表Query，K代表Key，V代表Value。相对于上面的seq2seq+attention模型，这里做了一个attention概念上的拆分，Q和K去计算权重，V去和权重做加权平均。而在seq2seq+attention中相当于Q,K,V是一个向量。

而Multi-Head Attention则是多个独立的Scaled Dot-Product Attention计算得到的结果合并然后再经过一个全连接层。如下图所示：

$$\text{MultiHead}(Q,K,V) = \text{Concat}(\text{head}_1,...,\text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$



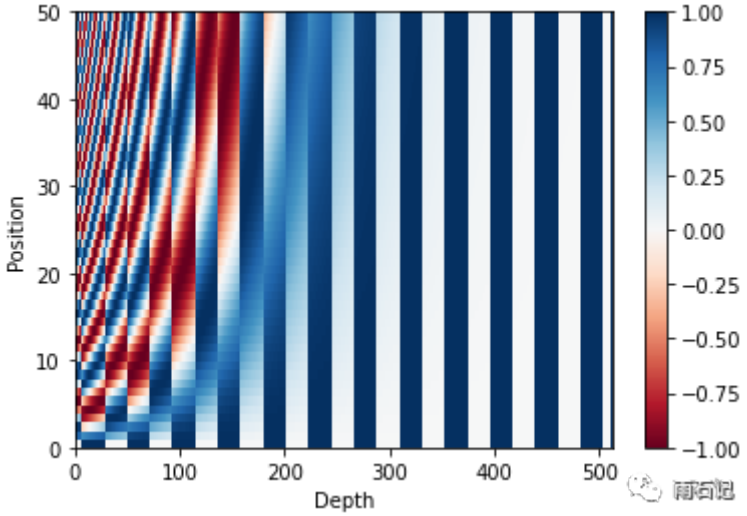
## Positional Encoding

因为用Attention替换了循环神经网络，所以原来循环神经网络里本能自带的位置信息就没有了，所以除了普通的随机初始化的embedding，我们还需要让模型知道每个输入的位置信息，最简单的方法就是给每个位置一个固定向量，然后让这个固定向量去和随机初始化的embedding去加和。原始论文中采用的是正弦余弦交叉混编的方式去做的：

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}})$$

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

这样给每个位置生成的向量如下，纵轴是位置，横轴是维度。

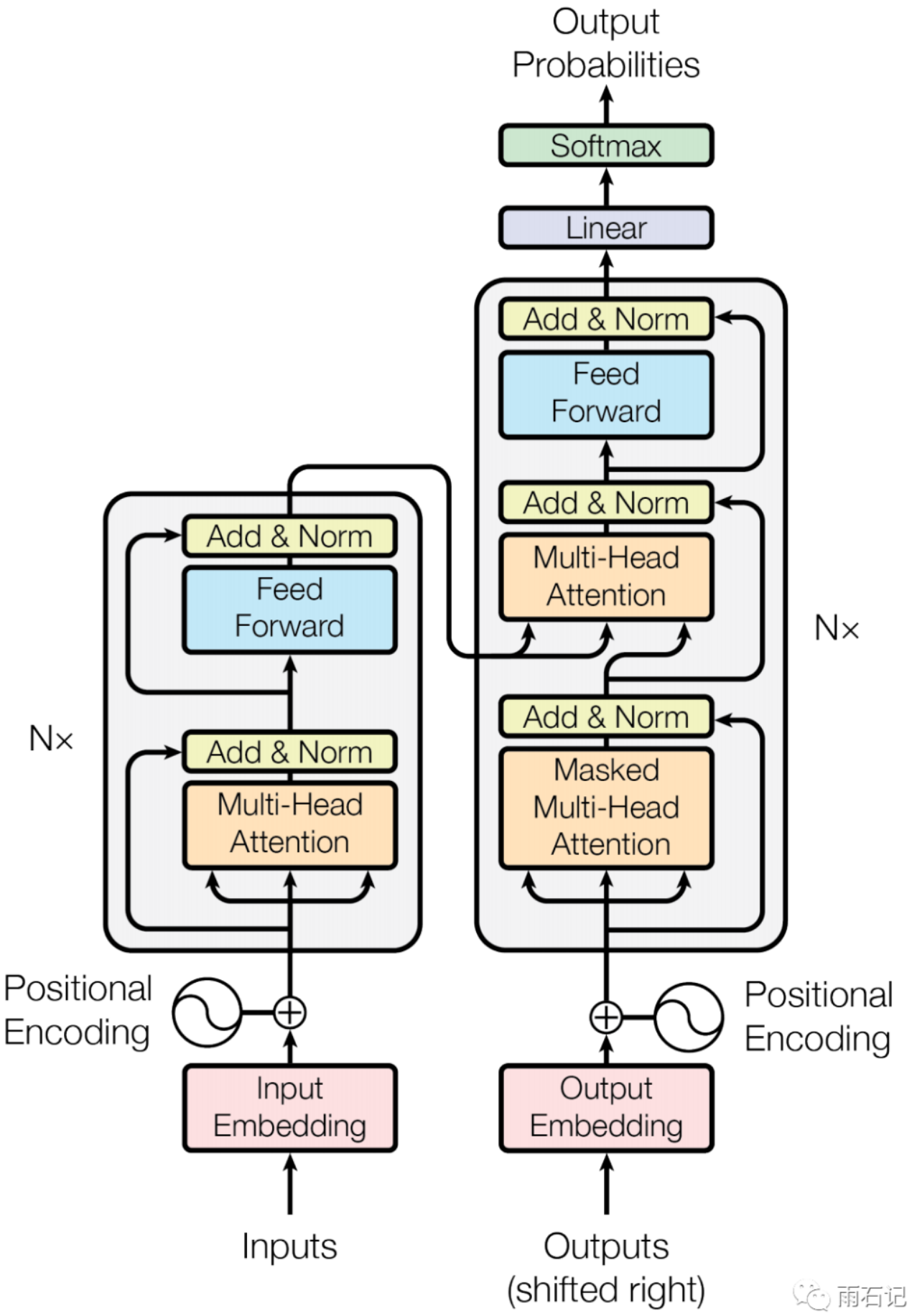



## Masked Multi-Head Attention

之所以需要mask，是因为在目标语言上，每一步是预测下一个词，所以在预测下一个词的时候不能让模型看到下一个词以及之后的信息，所以在处理目标语言的时候需要对attention做mask，attention本来是一个二维矩阵，即每个位置对每个位置的权重，做了mask后就相当于强制一半的值（二维矩阵的右上三角）为0。这就是mask的含义。

## Encoder与Decoder

上面几个小节介绍了各种子模块，现在我们再回到大图。


Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)
雨石记

继续提问：

> 问：Encoder和Decoder是怎么联系的呢？
>
> 答：Decoder的每一个block比Encoder的block多一个Multi-Head Attention，这个的输入的K,V是Encoder这边的输出，由此，构建了Encoder和Decoder之间的信息传递。

## 总结

以上就是对Transformer的解析，对于翻译问题，Encoder和Decoder都必不可少，但对于语言模型建模的问题，只需要Decoder端就够了（此时把中间的Multi-Head Attention给去掉)。

作为一个比较老的论文，其实网络上有很多资料来解释Transformer。参考[2]里有一个更清楚的说明Transformer计算过程的图解。参考[3]里则是Tensorflow2.0官方的Transformer实现。

## 思考

> 勤思考，多提问是Engineer的良好品德。

- 问题1：为什么要分成Q,K,V三个去做attention计算？
- 问题2：为什么要用多头？
- 问题3：试分析Transformer的性能瓶颈
- 问题4：只用Decoder端的话，Inference能不能增量？如何做到？
- 问题5：Encoder和Decoder都有多个block，每个里面都有attention，试推测不同block的attention会有怎样的差别？

欢迎关注，答案后续会发布。

## 参考

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. 一个详细的说明Transformer计算过程的文章 (http://jalammar.github.io/illustrated-transformer/)
3. Tensorflow2.0官方文档实现Transformer (https://www.tensorflow.org/tutorials/text/transformer/ )