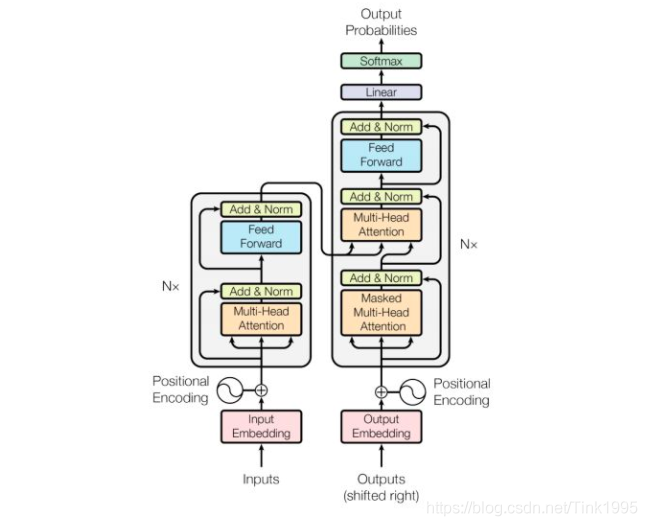
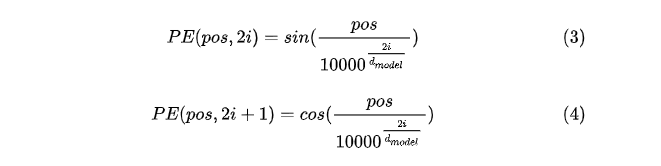
Transformer



Input：首先将文本转化为词向量（例如用one-hot编码），然后加入位置编码

2i和2i+1表示的是奇偶位置。位置编码使用的是叠加而不是拼接（因为效果差不多）。位置编码公式与三角函数加法有关：

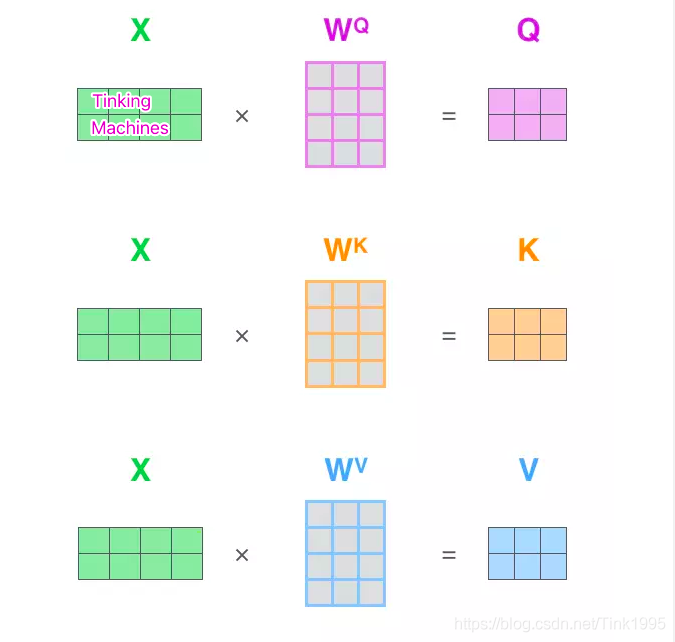


因此位置编码含有位置相对信息。

在训练时input和output同时接收source和target。例如input输入的是Tom chase Jerry，output接收的是“汤姆追逐杰瑞“。预测时input输入仍是Tom chase Jerry，output每次的输入是上一次的输出，例如，输入“”，输出“汤姆”，输入“汤姆”，输出“汤姆追逐”，输入“汤姆追逐”，输出“汤姆追逐杰瑞”，输入“汤姆追逐杰瑞”，输出“汤姆追逐杰瑞EOF”。

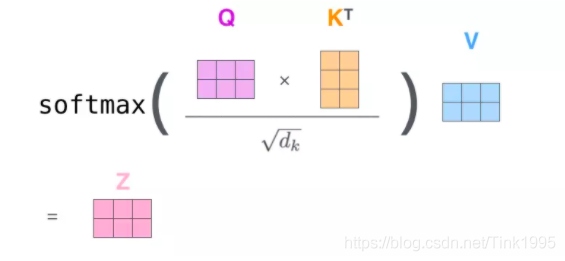
**Encoder：**

注意力机制中的QKV都是由输入词向量X得来的。如下图所示（输入为thinking machines，，，的列数都是64，行数是H和X相关，多头注意力机制中头的个数，见李沐Bert讲解24:00）：



Attention计算方法如下：

1. 用Q中每一个行向量点乘K中每一个行向量，即。
2. 将相关性得分归一化，即，是K中行向量维度（为了梯度稳定）。
3. 用softmax将矩阵score中每一个元素（单词与单词之间的关系）转化为概率分布。

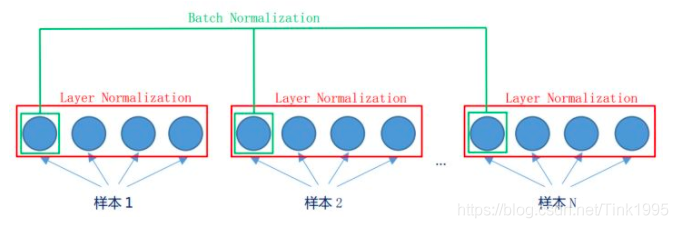


在transformer里用了8组，，得到8个Z，然后将这8个Z拼接。这8组，，是训练得到的。

经过注意力层后，Z需要经过Add＆Normalize。

Add就是Z=Z+X。这和resnet是一个道理，用来解决神经网络退化。

Normalize使用的是Layer Normalize而不是Batch Normalize。LN是在同一个样本中不同神经元之间进行归一化，而BN是在同一个batch中不同样本之间的同一位置的神经元之间进行归一化。而单独分析词向量的某一维度是没有意义的（例如onehot编码）。



Feed Forward Network：

例如x（Z）是2\*64维矩阵，假设W1是64\*1024（bert中是4H 64\*4=256）的，那么W2应该是1024\*64的，b1是2\*1024的，b2是2\*64的，这样FNN(x)和x大小相同。这样的目的是将输入的Z映射到更加高维（2\*1024）的空间中，然后通过非线性函数ReLU进行筛选，筛选完后再变回原来的维度。

**Decoder：**

Decoder相对于Encoder而言多了mask部分。Mask分为两种：

1. Padding mask：由于每一批输入的序列长度不一样（例如“汤姆”和“汤姆追逐”），因此要在较短的序列最后填充0。如果填充的0太多，就只取左边的部分。可以给多余的0加上负无穷，这样经过softmax这些位置会变成0。Encoder里也需要padding mask。
2. Sequence mask：t时刻的预测只能依赖于t-1时刻的输出，因此需要把t时刻及以后的内容掩盖掉。这是用在测试集上的。训练集上可以看到完整的target。

Attention：

Decoder中的attention的Q是经过mask的词向量，K，V都来自于Encoder的**最后一层**。

输出：

输出首先得到的是词表中的词的预测概率，然后选取概率最大的作为t时刻的预测值。