**Transformer**

**Embedding**

将每个token转换为维的词向量

输入大小：Batch×

输出大小：Batch××

其中为待翻译语言最大句子长度

#实际翻译句子时，Batch=1，为清晰起见，本文档下文忽略Batch

**位置编码(Positional Embedding)**

Embedding后的token并不携带位置信息，故引入位置编码

pos为token在单词中的位置，pos[1, length\_seq]

2i/2i+1对应词向量的分量，i[0,256]

将位置向量与相应的词向量相加，得到神经网络输入

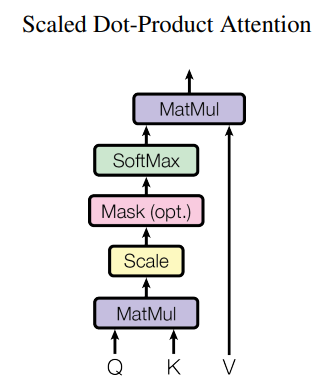
**缩放点积注意力(Scaled Dot-Product Attention)**

输入向量首先通过线性变换得到查询矩阵

其中为目标语言最大句子长度，第一种情况，均来自编码器或解码器内部，是自注意力；第二种情况来自解码器，来自编码器，是编码器-解码器注意力

#下面就第二种情况讨论，第一种情况同理

其中，，，，的每行对应一个单词



算出后就可以计算self-Attention的输出

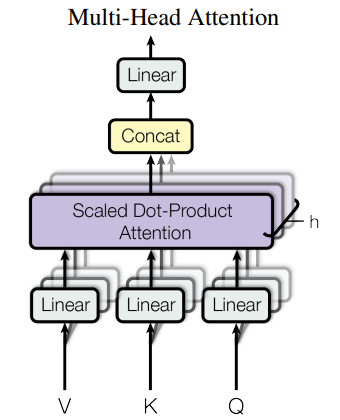
解释：#暂且忽略

表示与的相关性

经softmax按行归一化后，第表示目标语言单词和原语言单词的相关性分布

用对各个值向量加权再求和得到self-Attention的输出，越大，说明受影响越大，也可认为受影响越大，应该重点关注

**多头注意力(Multi-Head Attention)**



可以采用多头注意力的方式计算出多组

再计算各个

其中，，

计算出各个后可计算MultiHead的输出

其中，，

**逐位置计算的前向网络层(Postion-wise Feed Forward Networks)**

其中，，，

第一个线性变换将维度变大，第二个线性变换将维度还原

逐点计算指的是对每个position作同样的运算，即每一横行都用同一套参数计算，最终拼接成

**Add&Norm**

先将Sublayer层与位经过该层的样本与经过该层之前的样本相加，即做残差连接，Sublayer可能是MultiHead Attention或FFN

再对其和进行层归一化，即求

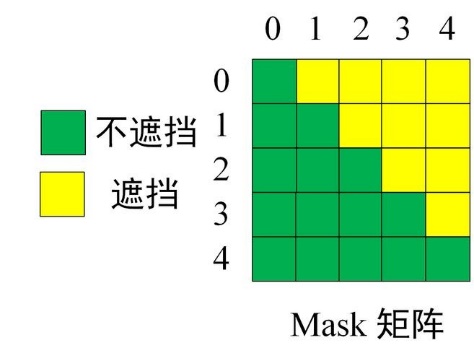
层归一化：

对于每个样本，先拉平成一维向量

求出结果后再reshape到，其中

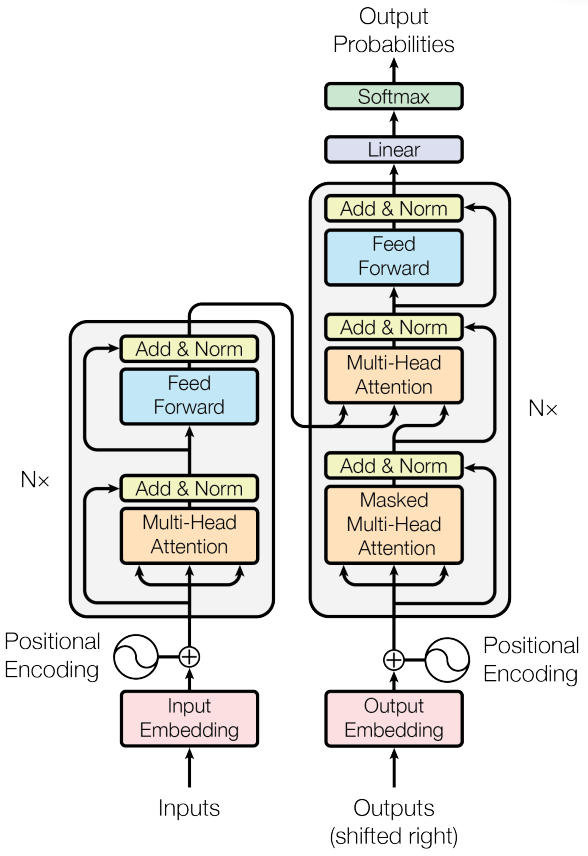
**Mask**

解码器运行时，为保证每个token只看到后面的token，采用掩码



在解码器第一个Multi-Head Attention计算出output的之后，将遮挡元素置为，不遮挡元素不变，得到Mask，再做softmax并与矩阵相乘

**整体流程**



**第一步：**获取待翻译句子句子中每个单词的词向量并与位置编码相加，得到单词的表示向量

**第二步：**将得到的表示向量传入Encoder中，经过个结构相同的Encoder block后可得到所有单词的编码信息矩阵，每个Encoder block的输出维度与输入维度完全一致

**第三步：**将Encoder输出的编码矩阵E传入第一个Decoder block第二个MutiHead Attention中，解码器收到编码矩阵E后，即将目标句子表示向量输入

**第四步：**E在解码器中经线性变换得，与解码器生成的Q做缩放点积，后面的5个Decoder block每个MutiHead Attention的均来自解码器自身，经过这5个Decoder block后得到解码器的输出D，每个Decoder block的输出维度也与输入维度完全一致

**第五步：**得到解码器输出后经过一个线性层，映射到目标语言词表的维度，即得到P，并用softmax归一化