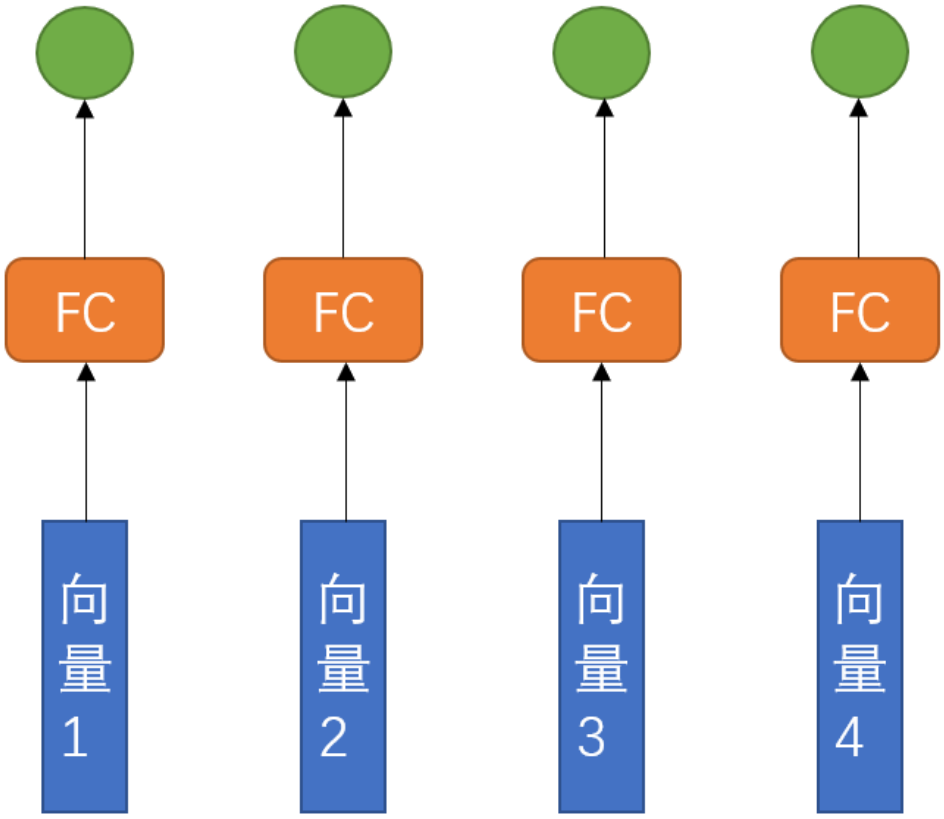
Self-attention和Transfomer

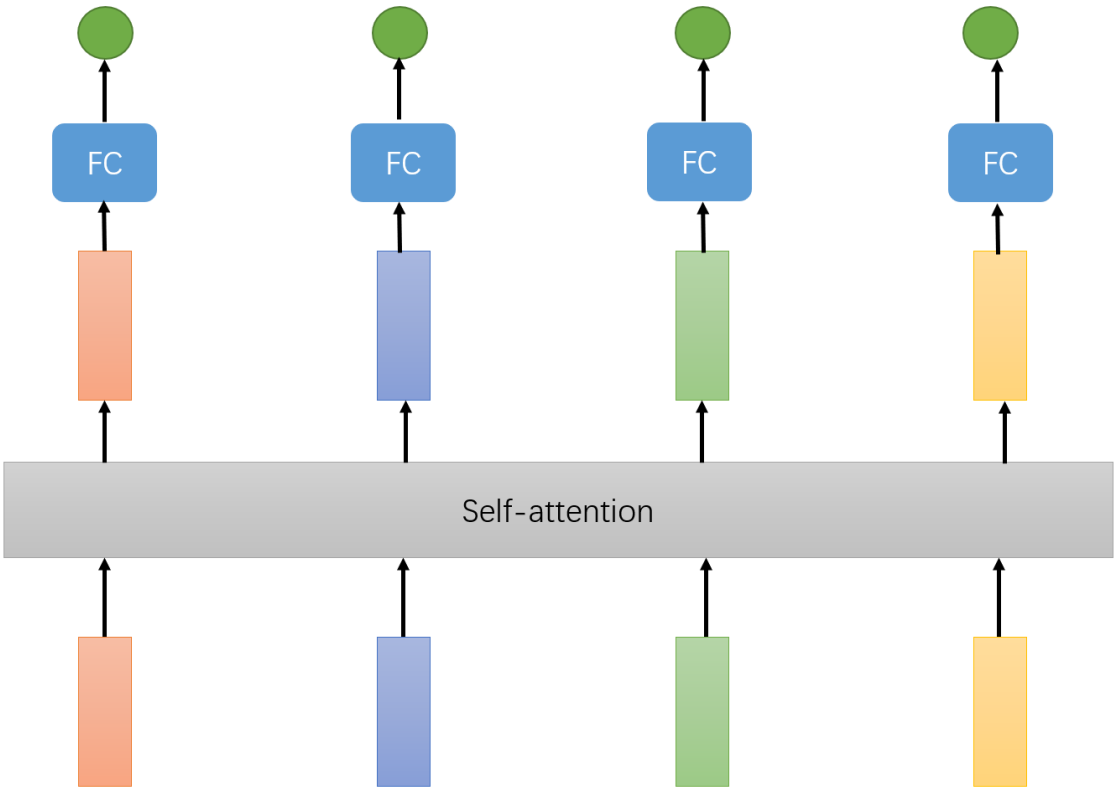
1.Self-attention

- 解决问题:Sequence to Sequence（序列输入到序列输出）
  输入是一组向量，同时组里的向量数目会发生变化。
  输出可能性有三种:输入长度和输出长度相同（词性标注）(Sequence Labeling)，一整个序列只需要一个输出，机器自行决定要输出多少(seq2seq)（机器翻译）。
- 举例:
  文字处理。每个句子的长度不一样（如果把句子中每个词都描绘为一个向量）。

如果使用全连接网络进行挨个输出，则无法考虑到序列本身带来的信息。
例如:如果在进行词性标注任务时，假设向量1和向量3都是一个词，但是因为位置导致语义词性不同，但是使用简单的全连接网络会使得二者输出相同，因此设计一种网络兼顾向量本身和序列本身带来的信息。



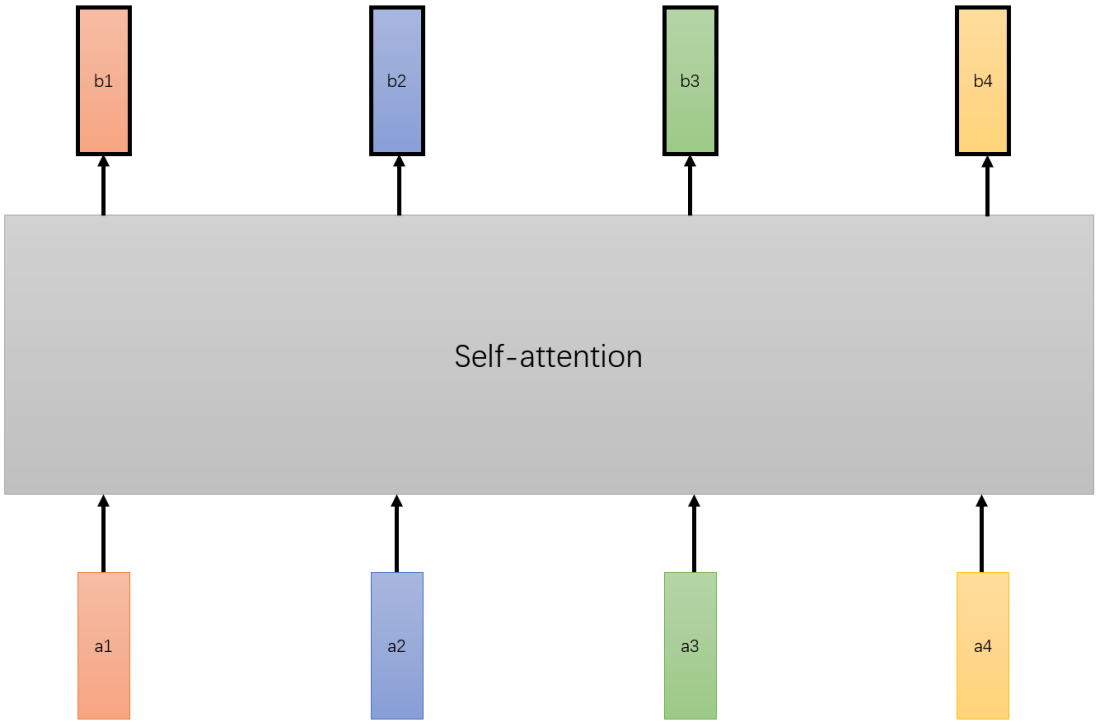
Self-attention一次性读入整个Sequence里的向量，从而在分析某个向量的时候考虑上下文(整个Sequence)的信息。



Self-attention输出的结果是考虑了整个句子的信息结合向量本身得出来的结果，同时，就输出的数量来说，输入多少向量，就会输出多少向量。

1.1模型架构

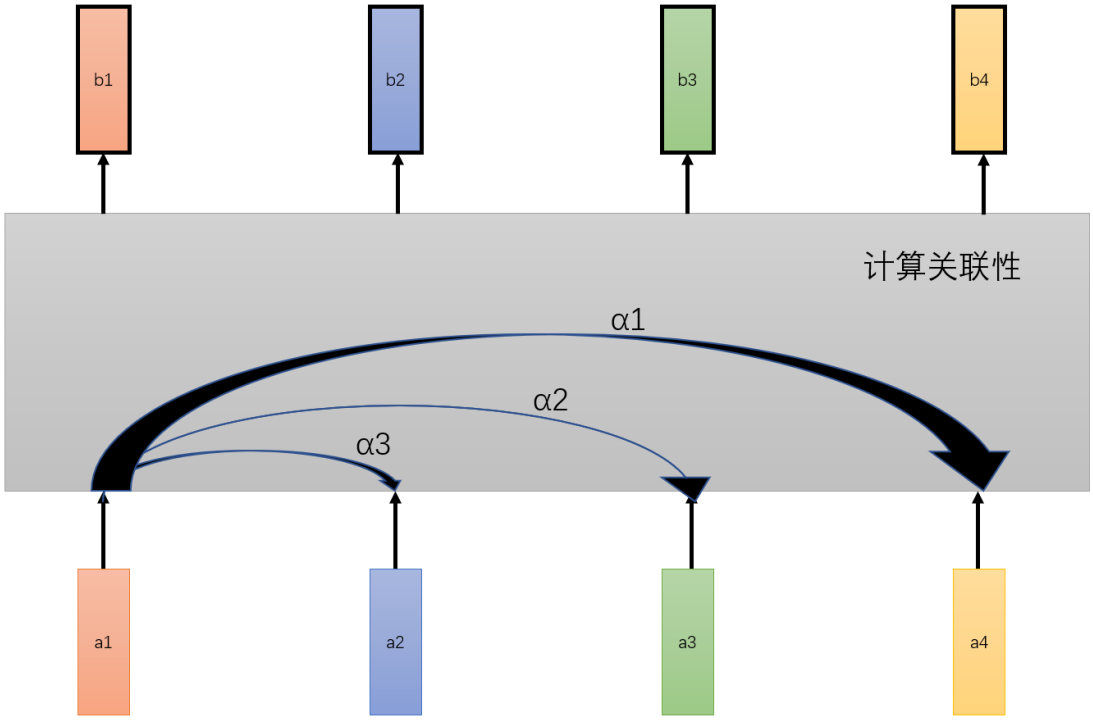
接下来描述Self-attention的模型架构:



如图所示，假设输入为一组向量[a1,a2,a3,a4]，输出为另一组向量[b1,b2,b3,b4]。b向量是由其对应的a及考虑序列关系得出的结果。

Self-attention的计算步骤(以得到b1为例):

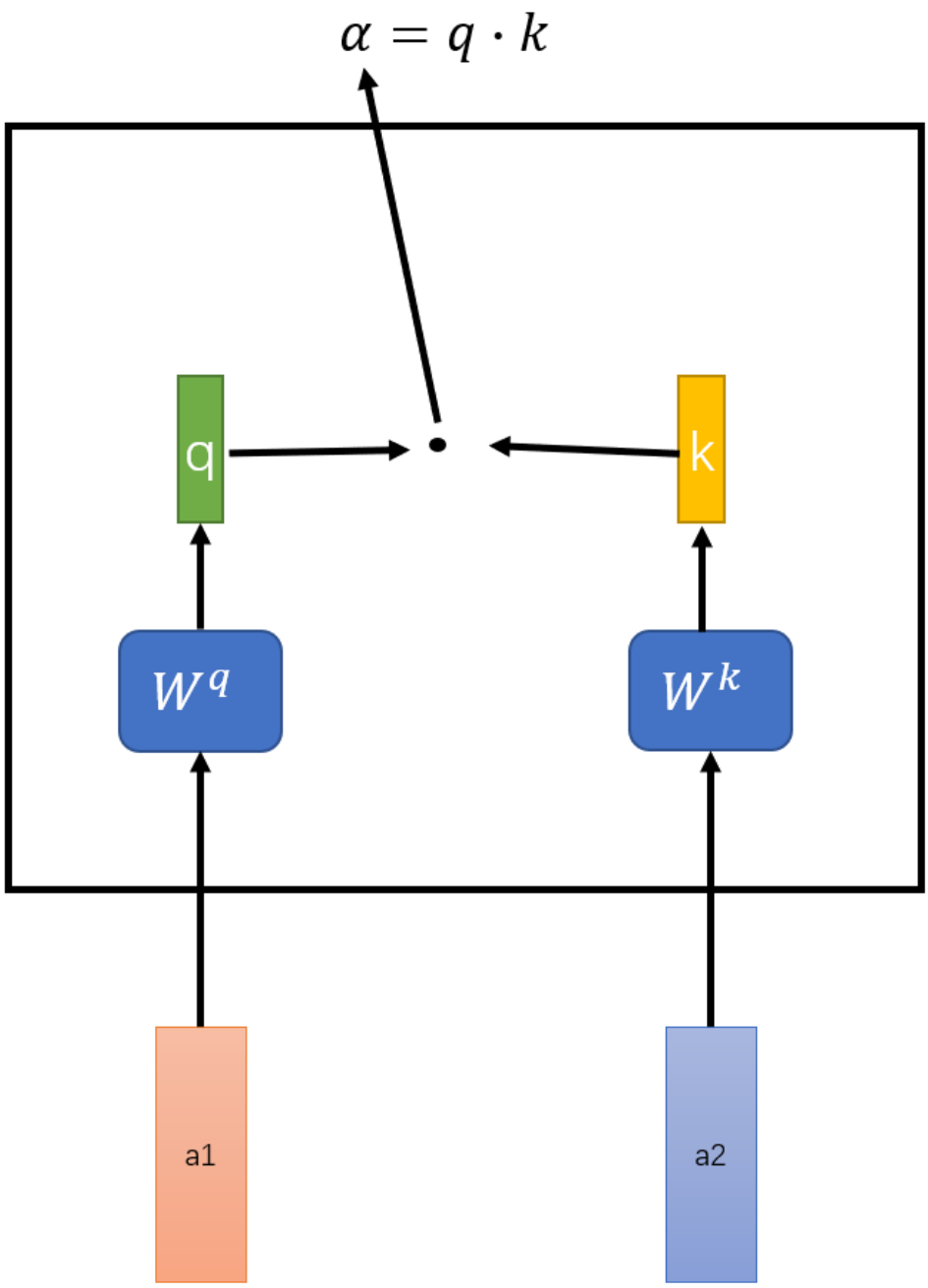
1.1.1计算a2,a3,a4和a1的关联性。



比较常见的计算两个向量关联度的方法是Dot-product和Additive。

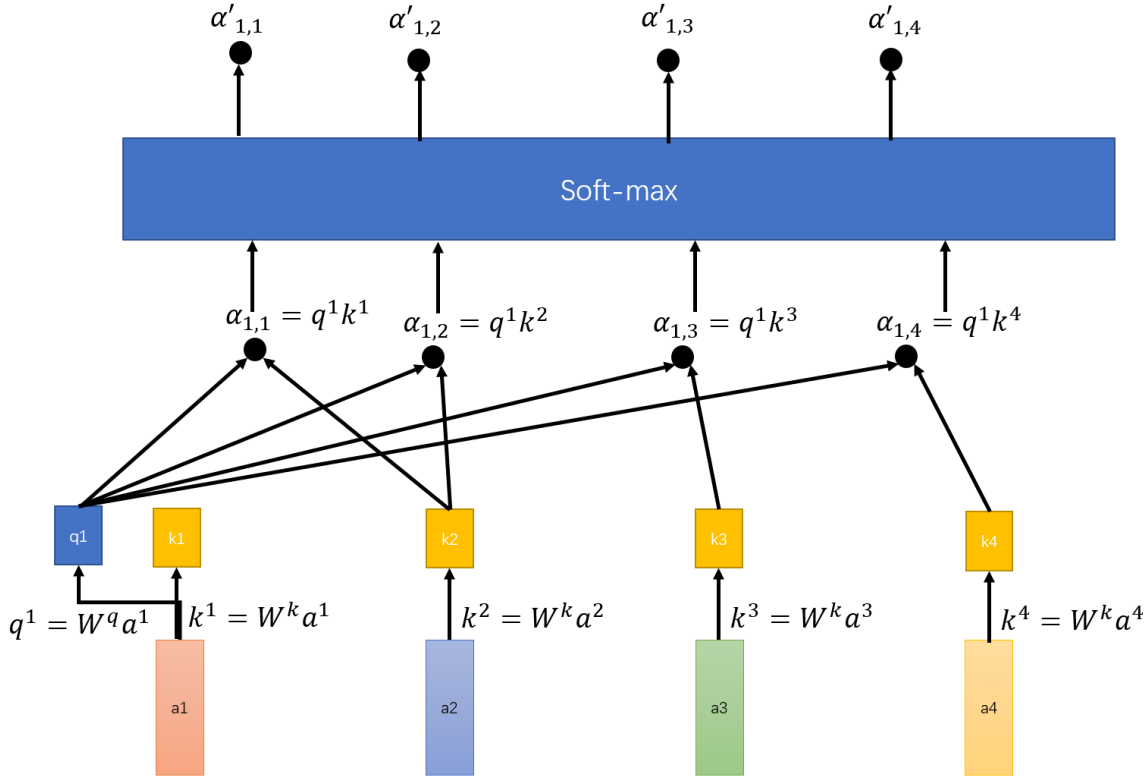
在本文只介绍Dot-product。

如下图，首先将输入的两个向量分别乘以矩阵W^q和W^k得到向量q和k，再对q和k做点积，$\alpha = q \cdot k$，得到两个向量的关联值α



一般在实践中，为了批量得到a1和a2,a3,a4之间的α，通常会采用如下的方式:
(1)将a1成矩阵W^q得到向量q1，然后用矩阵W^k一词乘a1,a2,a3,a4得到k1,k2,k3,k4.
(2)计算q1和k1,k2,k3,k4的关联程度，得到$\alpha_{1,1}, \alpha_{1,2}, \alpha_{1,3}, \alpha_{1,4}$

(3)将得到的关联结果送入softmax函数(也可以换其他激活函数)，得到$\alpha'_{1,1}, \alpha'_{1,2}, \alpha'_{1,3}, \alpha'_{1,4}$



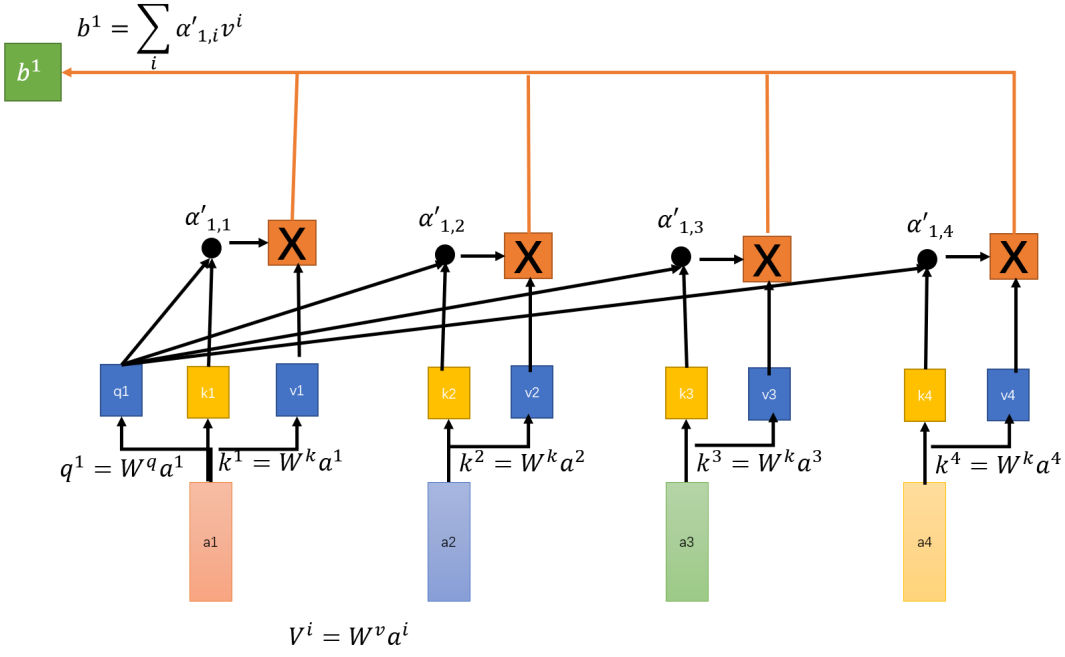
注:关联值 α' 也称为attention score

1.1.2抽取重要信息

经过上一步后，我们已经得到了向量序列中各个向量和向量a1的关联程度，接下来，通过这个关联结果，来抽取重要信息。

(1)首先针对a1,a2,a3,a4；我们使其每一个乘矩阵W^v得到新的向量v1,v2,v3,v4。

(2)然后使得每个v1,v2,v3,v4分别乘其对应的$\alpha'_{1,i}$，然后将乘积结果相加，得到b1。即:$b^1 = \sum_i \alpha'_{1,i} v^i$。二者关联越大，对b1的影响就越大。



经过上述步骤，我们可以从一整个sequence得到b1,b2,b3,b4。

如果从矩阵乘法的角度来看，上述两个步骤的结果是这样的:

- 计算q^i:
  $q^i = W^q a^i$
  $[q^1, q^2, q^3, q^4] = W^q [a^1, a^2, a^3, a^4]$
  设$I = [a^1, a^2, a^3, a^4]$,$Q = [q^1, q^2, q^3, q^4]$
  则有:$Q = W^q I$
- 计算k^i:
  $k^i = W^k a^i$
  $[k^1, k^2, k^3, k^4] = W^k [a^1, a^2, a^3, a^4]$
  设$I = [a^1, a^2, a^3, a^4]$,$K = [k^1, k^2, k^3, k^4]$
  则有:$K = W^k I$
- 计算v^i:
  $v^i = W^v a^i$
  $[v^1, v^2, v^3, v^4] = W^v [a^1, a^2, a^3, a^4]$
  设$I = [a^1, a^2, a^3, a^4]$,$V = [v^1, v^2, v^3, v^4]$
  则有:$V = W^v I$
- 计算α:
  $[\alpha_{1,1}, \alpha_{1,2}, \alpha_{1,3}, \alpha_{1,4}] = [k^1, k^2, k^3, k^4] \cdot q^1$
  $A' \leftarrow A = K^T Q$
- 计算b^i
  $B = V \cdot A'$

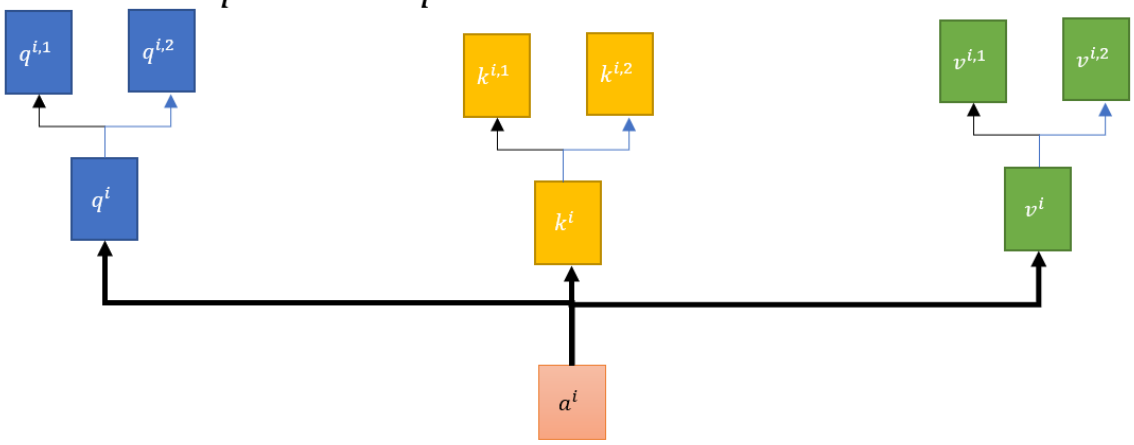
- 总结
  $Q = W^q I$
  $K = W^k I$
  $V = W^v I$
  $A = K^T Q$
  $A' \leftarrow A$
  $O = V A'$
  所以需要学习的参数，只有W^k, W^q, W^v。

1.2 Multi-head Self-attention

当我们在做Self-attention的时候，我们是在用$q^i = W^q a^i$来找当前向量和序列向量之间的相关性。
Multi-head Self-attention 出于这么一种考虑，即一个q无法包含当前向量和序列向量之间的所有相关信息，因此提出了使用多个q来包含当前向量和序列向量之间的所有相关信息。
因为q变多了，所以此时的k和v也应该变为多个。

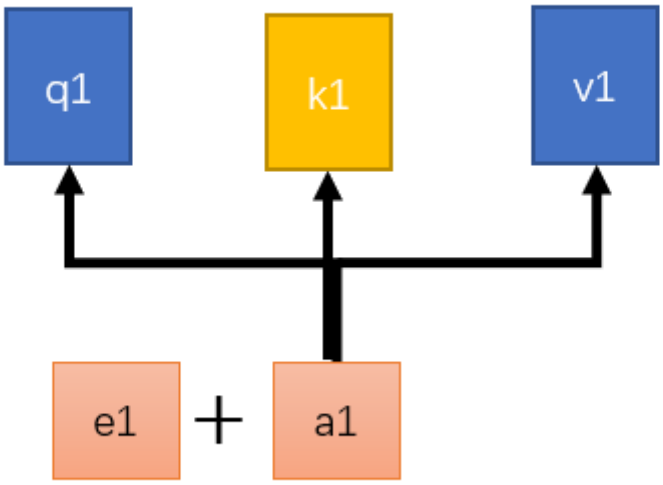
如下图所示(此处只有两个head):

$$q^i = W^q a^i$$
$$q^{i,1} = W^{q,1} q^i$$
$$q^{i,2} = W^{q,2} q^i$$



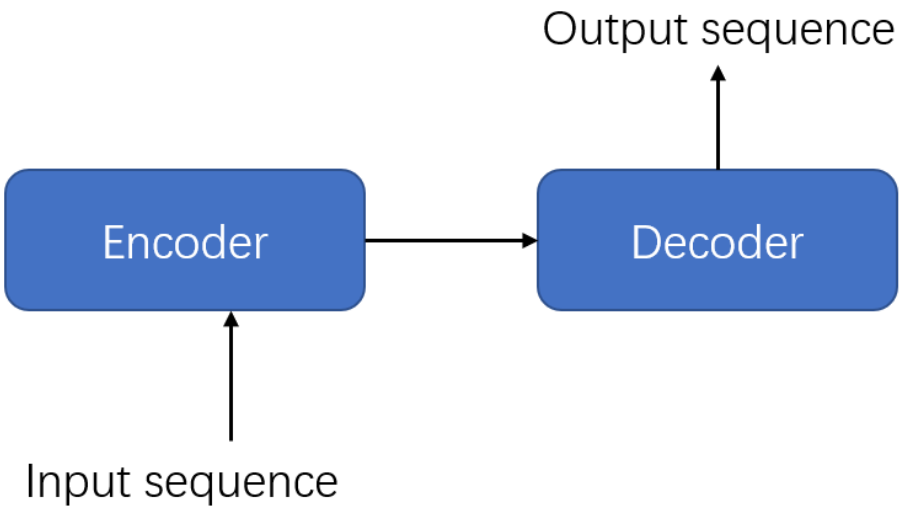
1.3 位置编码

到目前为止，Self-attention仍然没有考虑到序列信息(因为attention score)只计算了相关性。
解决方法:位置编码(Positional Encoding)
对位置进行编码，生成e^i，然后加上输入一起作为输入即可。



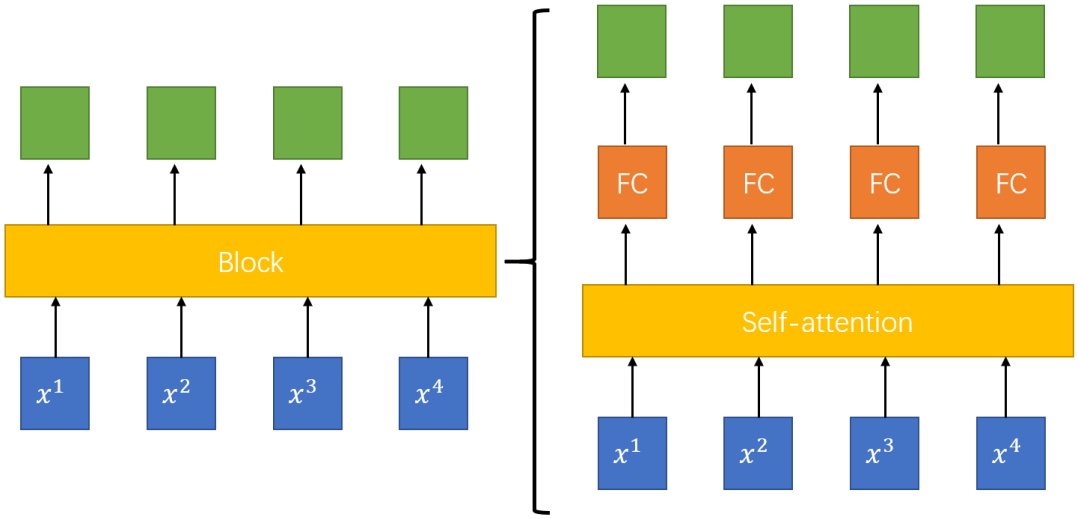
2.Transfomer

常用于处理Seq2Seq问题，即模型输出多少长度是由模型自己决定。例如机器翻译和语音辨识任务。
Seq2Seq一般架构:

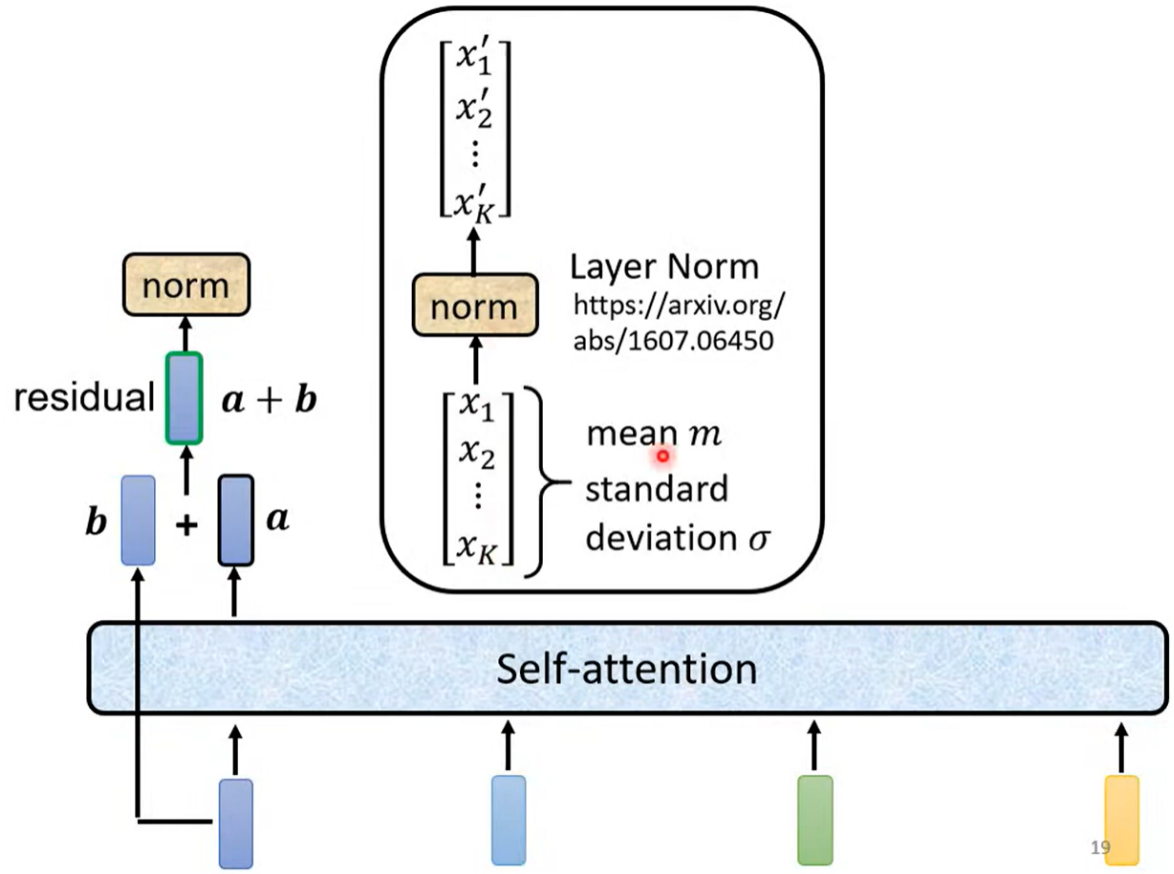


2.1 Encoder

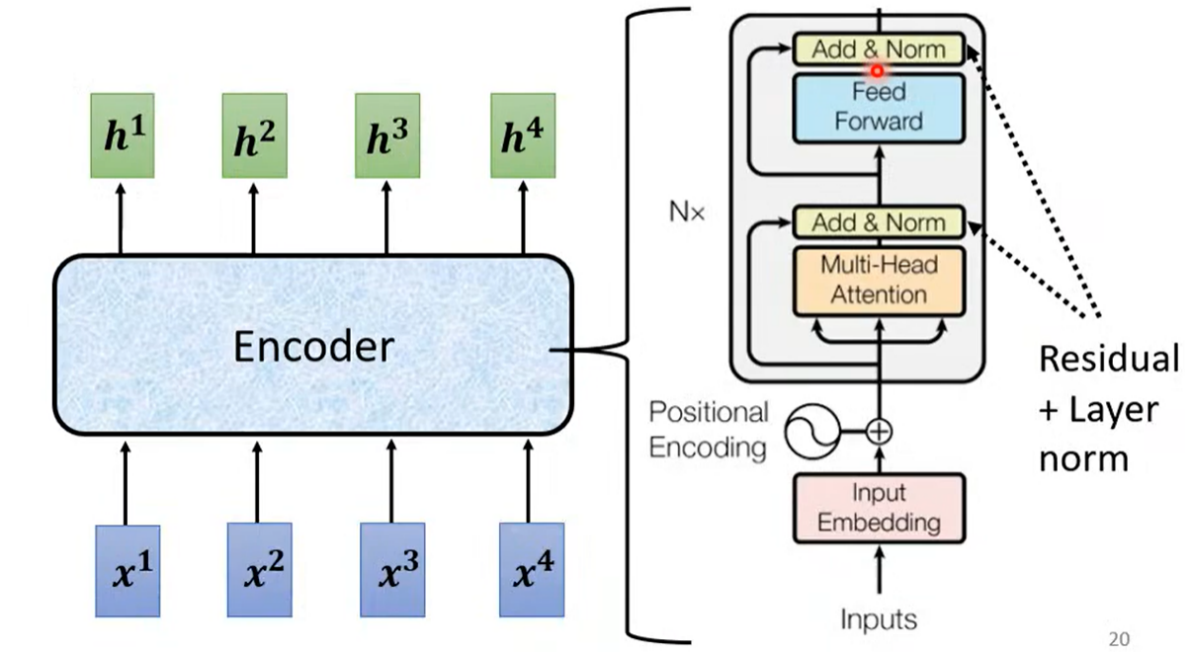
Input是sequence，Output也是同长度的sequence。



解释其中的一个block:
在Transfomer里，Self-attenion里，输出的b不只是Self-attention的计算结果，输出b应该是b和原向量a的加和(residual)。



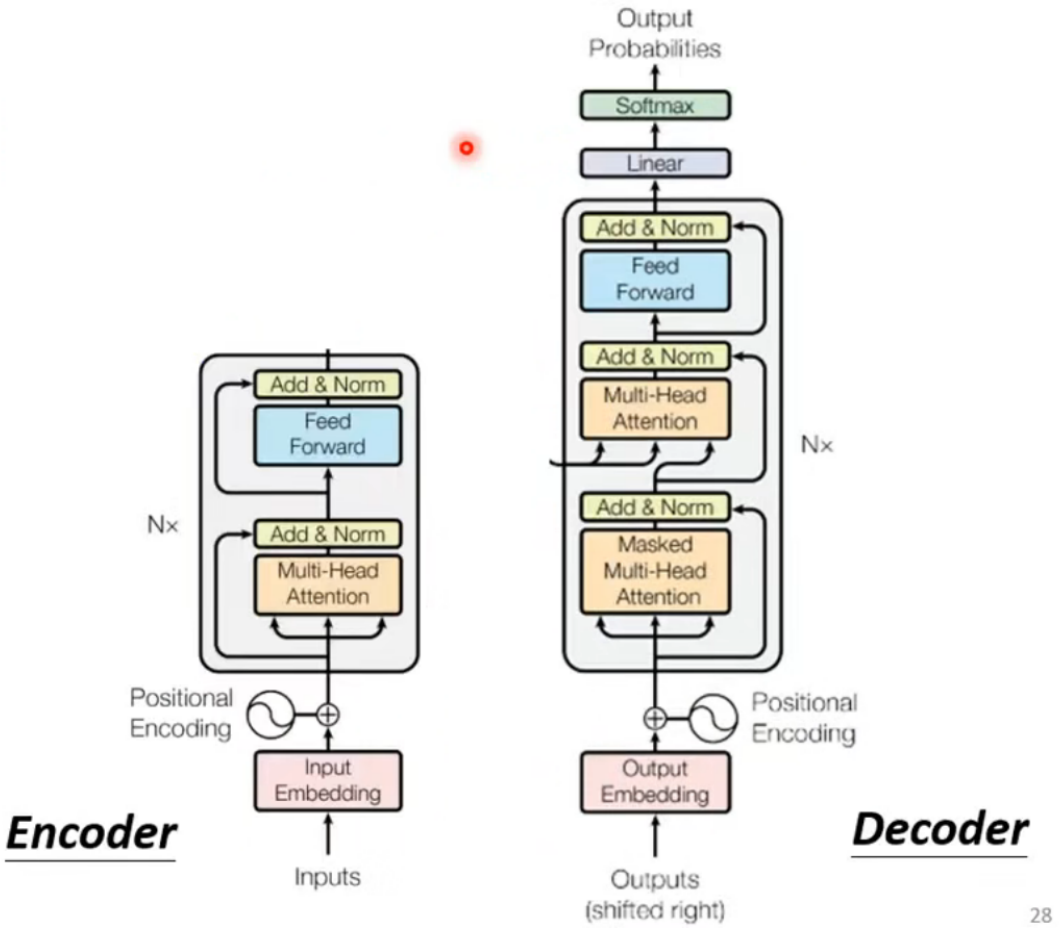
residual完成后，将输出结果送入Norm进行标准化，标准化完成后输入全连接网络(FC)。
FC的结果也需要进行residual，结果也要送入Norm一次，这样才是完成了一个Block的输出。



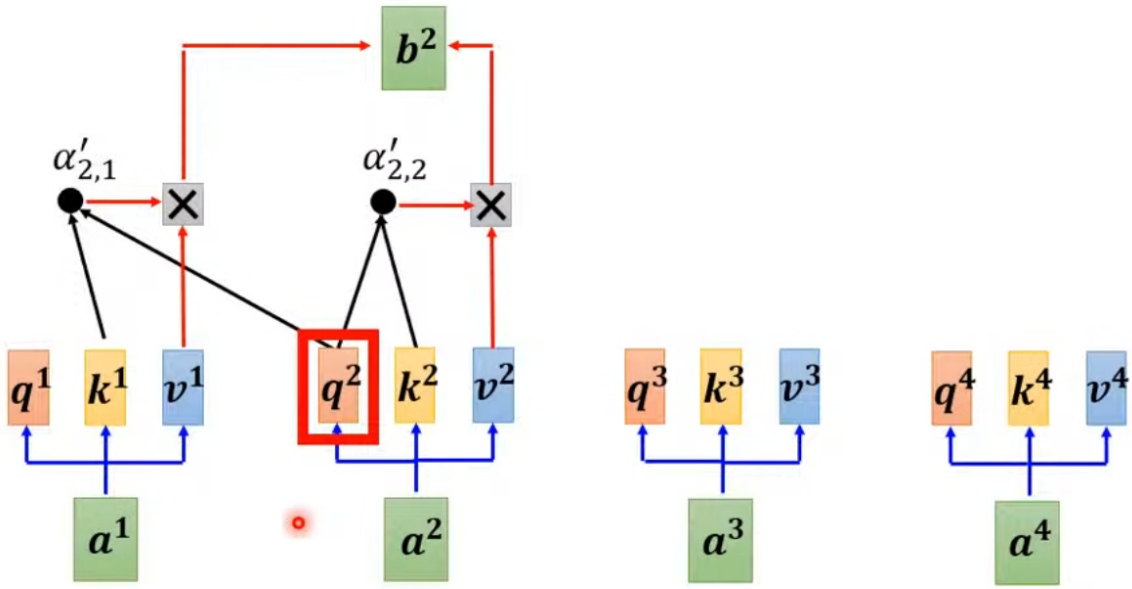
注意这里的attention是Multihead的。

2.2 Decoder

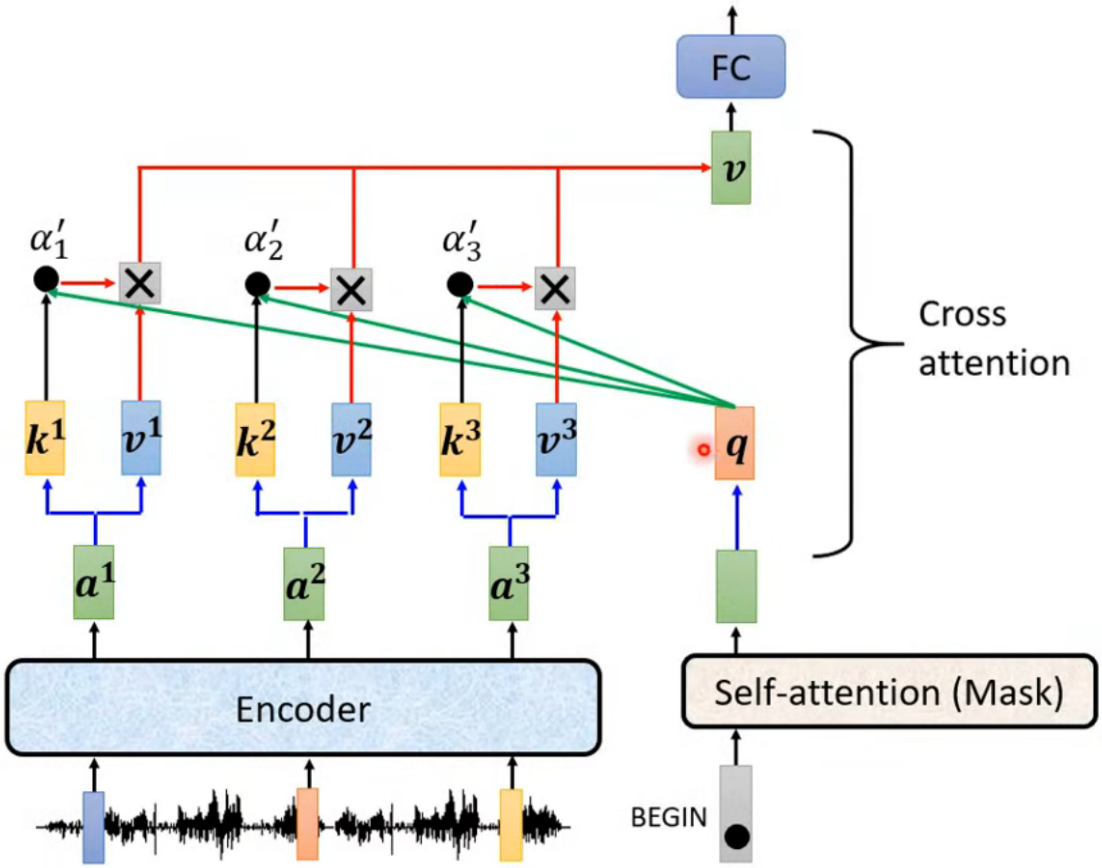
Decoder看到的输入是前一个时间点其本身(Decoder)的输出。
以下为Decoder的结构:



可以看出，Decoder结构和Encoder大致相同，但是Decoder的Self-attention比起Encoder中多了一个Masked。
意思是Self-attention只能查看其之前的数据的信息。



如图，在计算b2的时候，只考虑a1,a2的信息。
回顾最开始的目标，Transfomer的目标不仅是要模型产生输出，同时还要模型自行决定输出数据的长度，这里的实现方式是使得Decoder可以输出一个END作为特殊符号，表示输出结束。
如图为Decoder中Self-attention的工作机制:



Decoder的每一层都得看一下Encoder的输出。