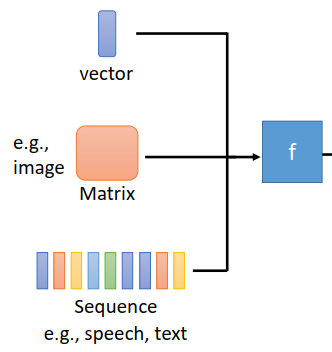
Self-attention

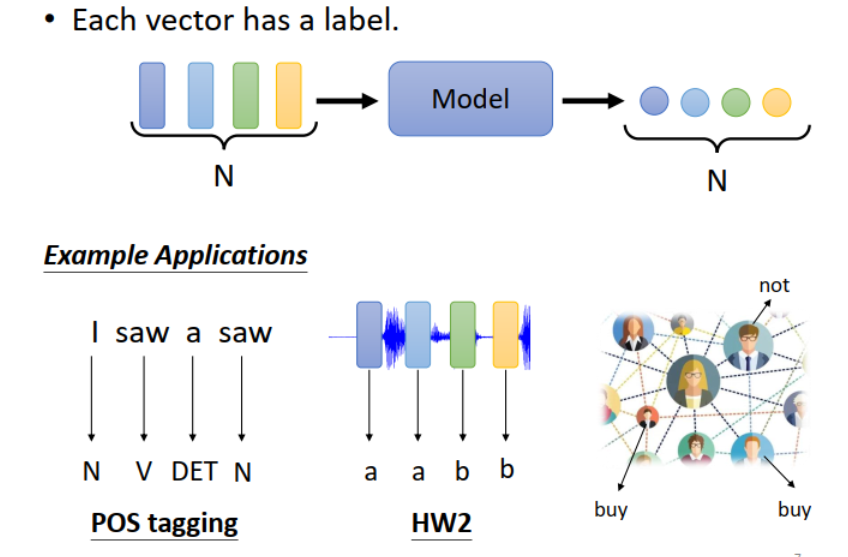
前面视频中的Model(Deep Neural Network)输入的每个sample是Vector或Matrix(e.g. image)的形式。其实Matrix可以看成是把Vector做了一个维度变换，从一维变成三维或更多维。这样看来，输入是Vector或Matrix时，每个sample的维度是固定的。当输入的一个sample是一组vectors时，不同sample包含的vectors个数可能不同。这样，sample的维度不再固定，是变化的。这种输入也被称为Sequence或vector set。



**输出形式：**

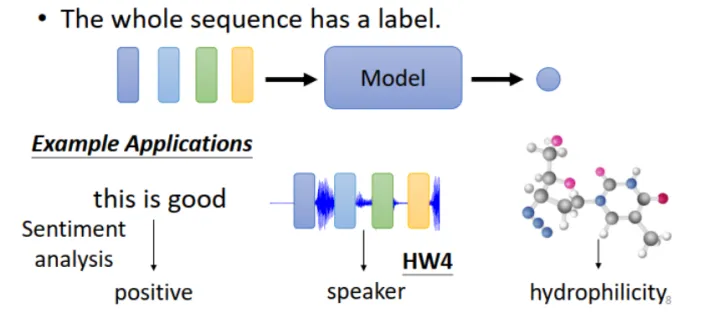
1.输出序列长度与输入序列相同。

每个 vector 都有一个对应的 label 输出，这样输出序列长度与输入序列长度相同。



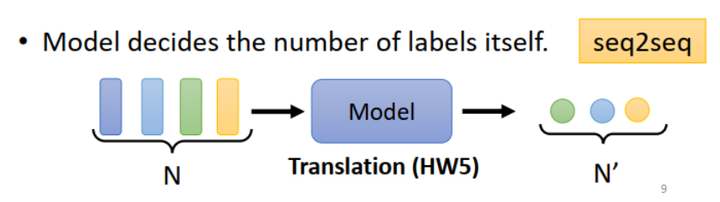
1. 输出序列长度为1。

The whole sequence has a label.整个序列输出一个label。



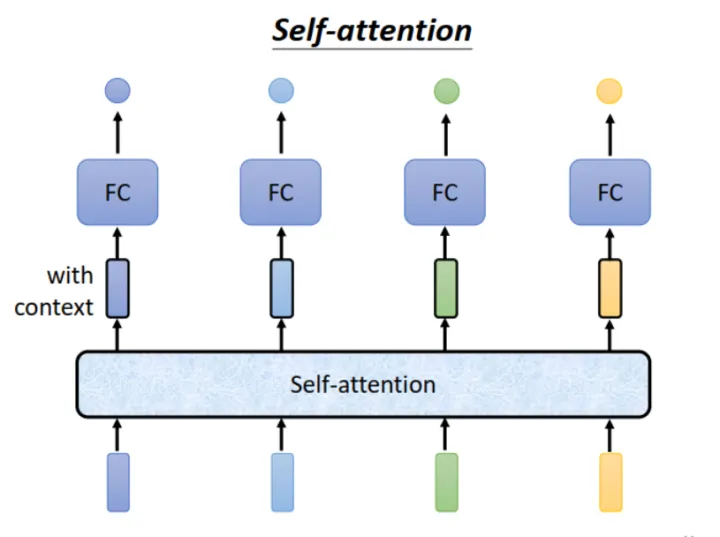
1. 模型决定输出序列长度。

由模型决定输出 label 的个数，也就是输出长度。

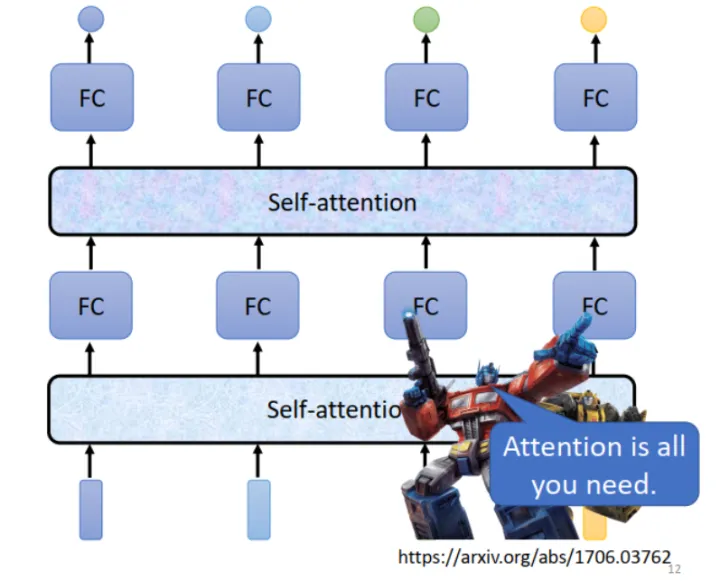


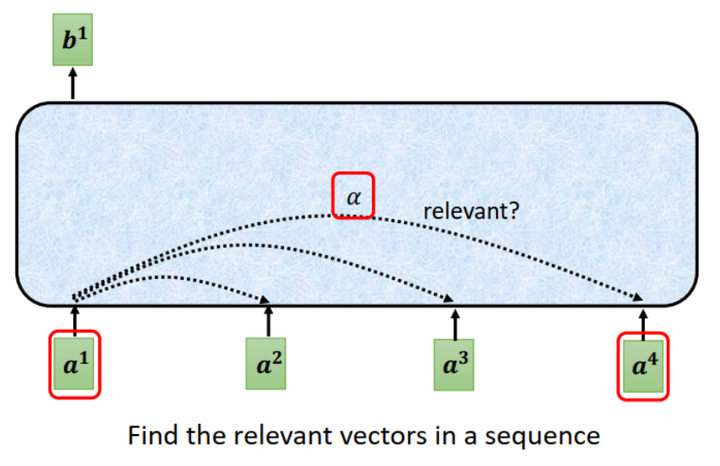
### **Self-attention 原理**

如下图所示，把 sequence (vector set) 输入 Self-attention 模块，输出的 sequence 中每个 vector 都带有 context 信息。然后把这些 vectors 送入 Fully Connected Network，和输入是 vector 一样来处理。

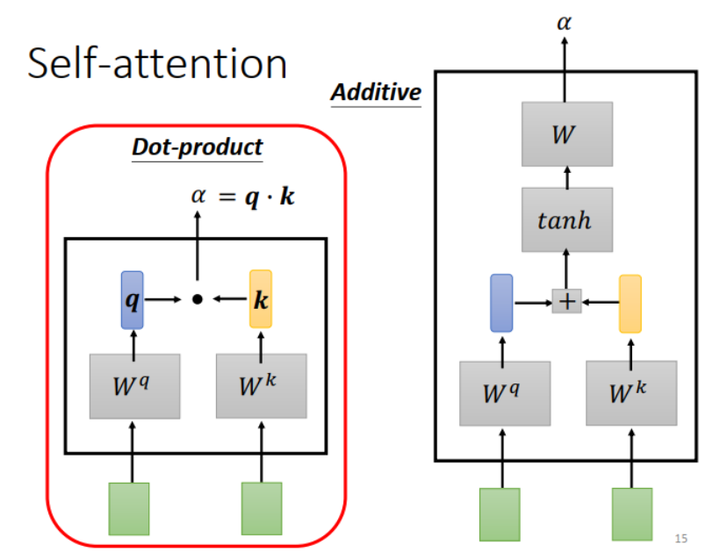


和 CNN 的 Convolutional Layer 一样，Self-attention+Fully Connected Layer 也可以叠几层，如下图所示。因此，self-attention 的输入层可以是 input 或 hidden layer。

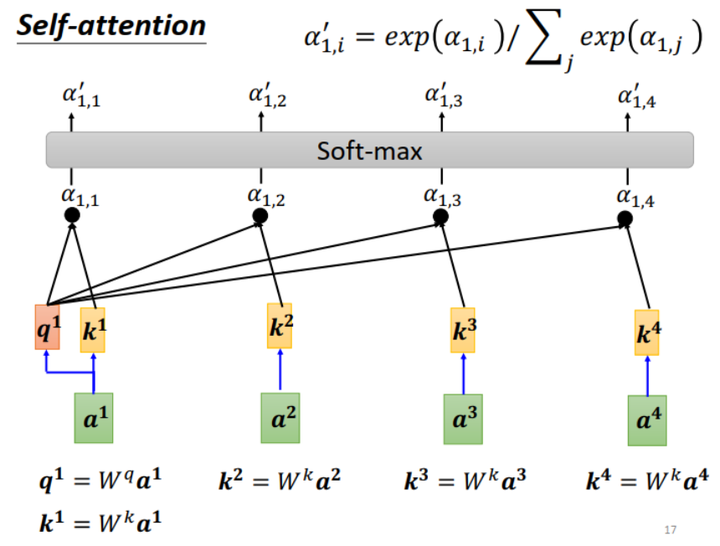
**而Self-attention 要做的就是寻****找 vector 之间的关系。如下图所示，对于某一个vector，求出其与 sequence 中其它 vectors 的相关性。我们可以用*a*来表示相关性。**



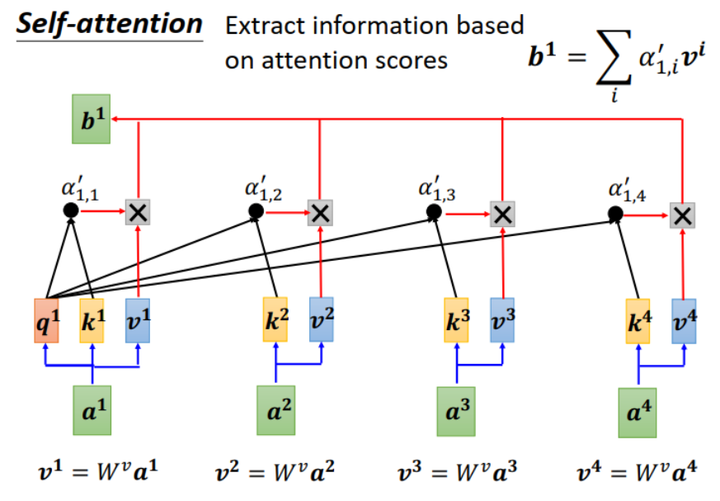
具体a计算原理如下图所示。



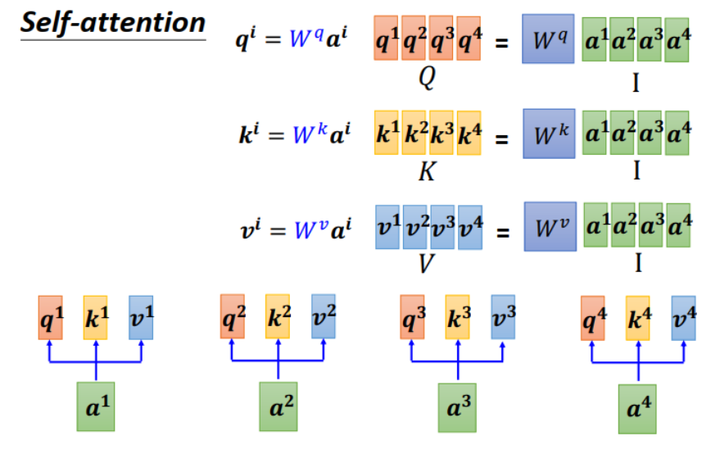
然后再进行归一化，归一化的选择可以有多种。



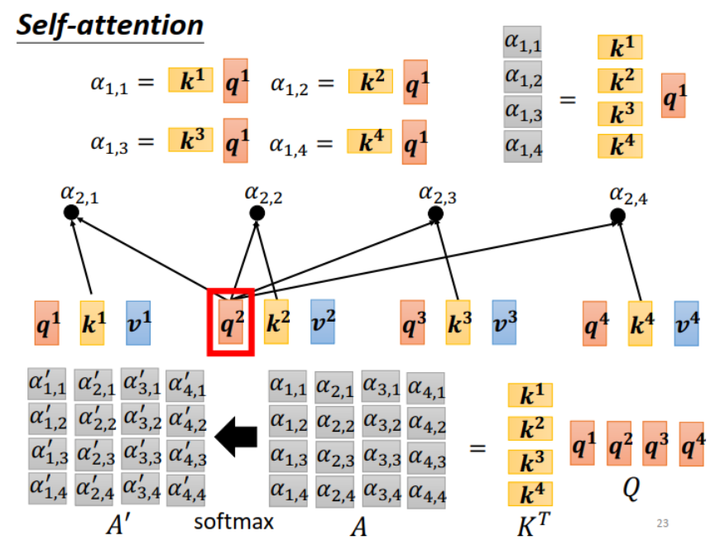
然后根据计算结果再来计算b

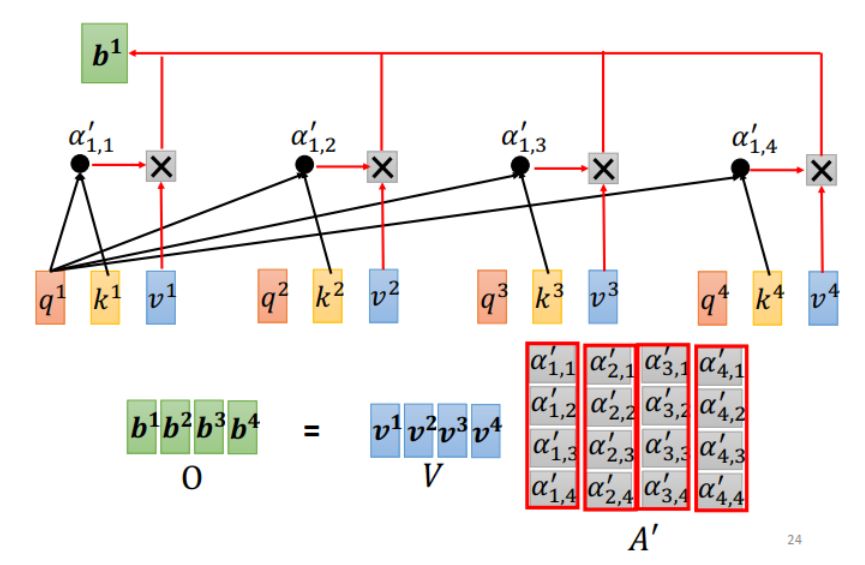


现在从个体转换到整体，来看看输入是 vector set (sequence) 时如何操作。

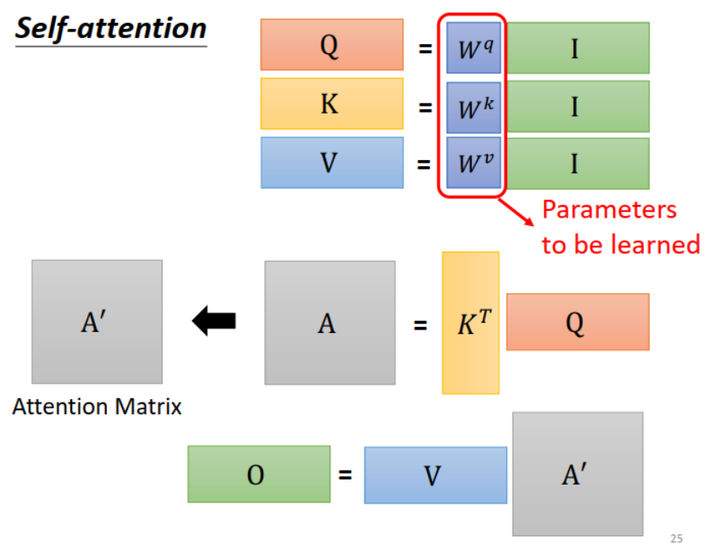


整体矩阵计算如下：

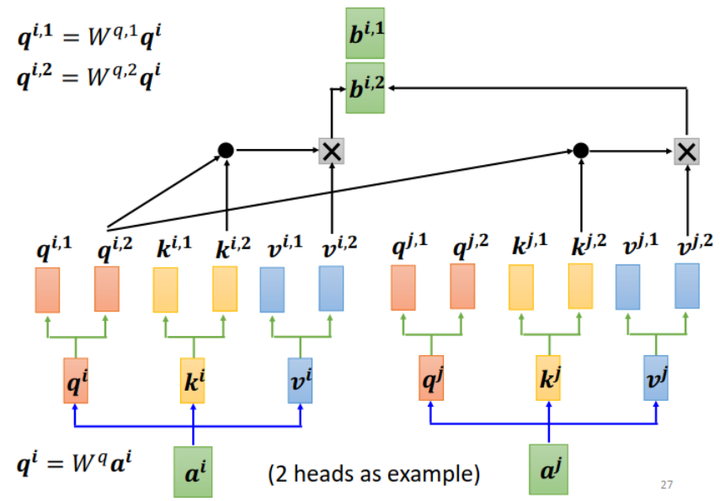




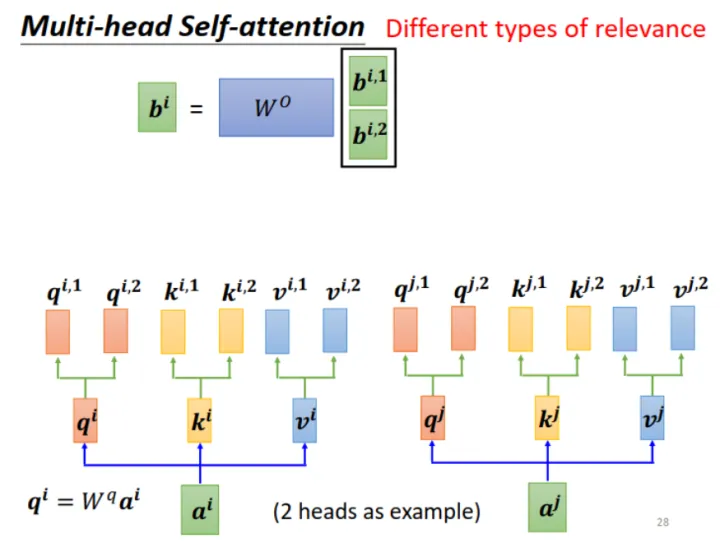
综合可得：

**Multi-head Self-attention：**

有时候，我们要考虑多种相关性，需要不只一种 self attention，于是有了 Multi-head Self-attention。如下图所示（图中为 2 heads 的情况）

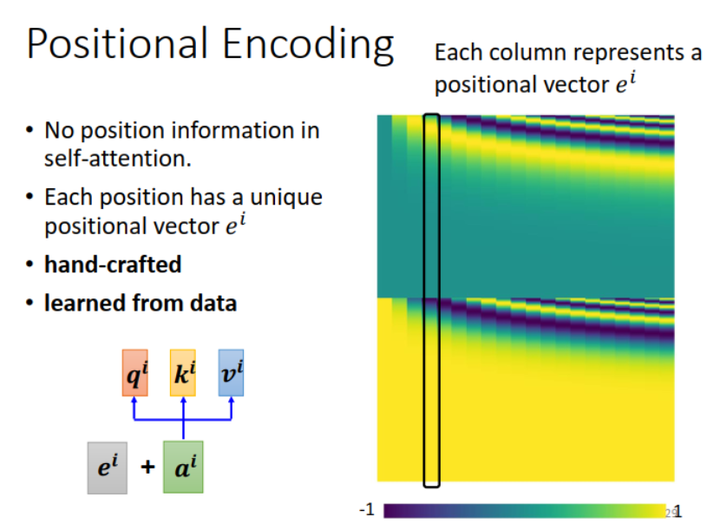


与单个的 self attention 相比，Multi-head Self-attention 最后多了一步：由多个输出组合得到一个输出。



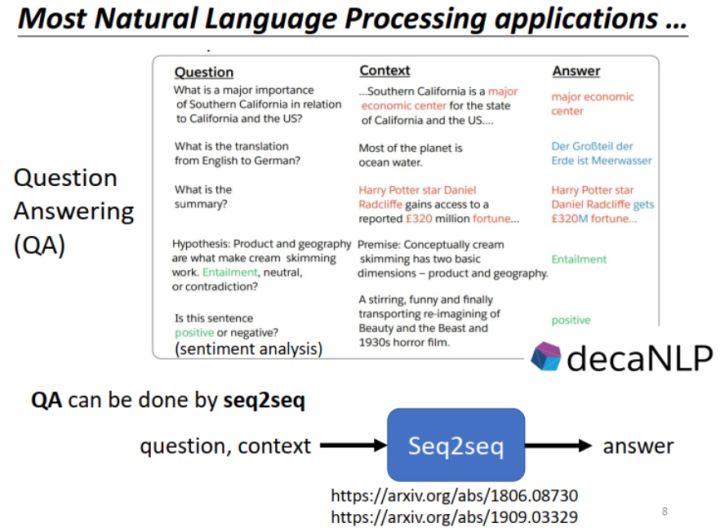
**Positional Encoding**

self-attention 没有考虑位置信息，只计算互相关性。比如某个字词，不管它在句首、句中、句尾， self-attention 的计算结果都是一样的。但是，有时 Sequence 中的位置信息还是挺重要的。所以将位置信息加入到输入中。



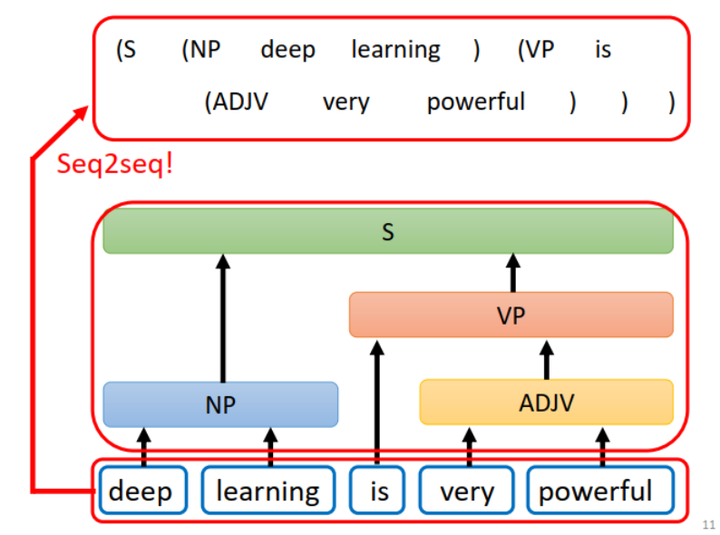
**Seq2seq**

Seq2seq模型应用广泛，尤其是在自然语言处理任务中，例如：语音识别，语音生成，语音翻译 (speech translation)，聊天机器人(Chat-bot)，等等。Seq2seq还可以用于解决Question Answering(QA)任务。其实很多问题都可以转化成QA来解决，例如：translation,sentiment analysis,abstract，等等。如下图所示，把目标当作Answer，文本数据当作Context，Context和Answer之间通过Question联系起来，就可以转换成一个Question Answering任务。而QA可以用Seq2seq解决。

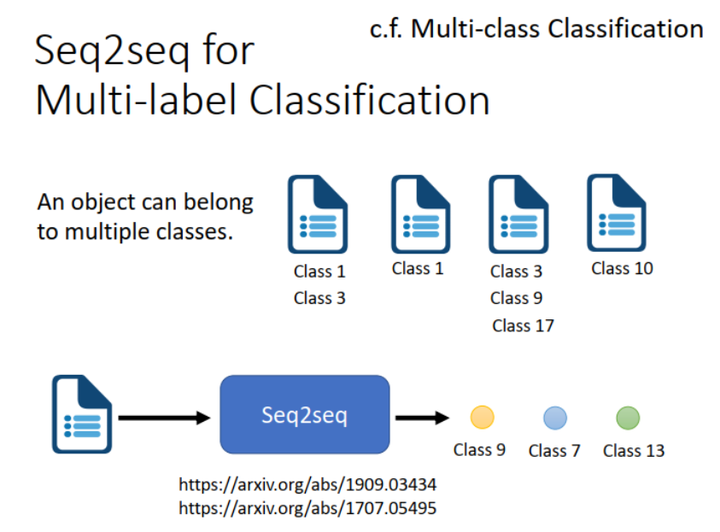


如此说来，大部分的NLP任务都可以用Seq2seq模型来解决。Seq2seq可谓是全能型选手。看到这里，你可能会想起，在讲 CNN（卷积神经网络）时介绍过，虽然Fully Connected Network是全能型选手，可以解决各式各样的问题，但是在图像处理上，它没有专门针对图像特性优化的CNN性能好。Seq2seq也是类似的情况，虽然它可以解决各种NLP任务，但在某个特定的语音或文本处理任务上，它的表现不如专门为任务设计的模型好。

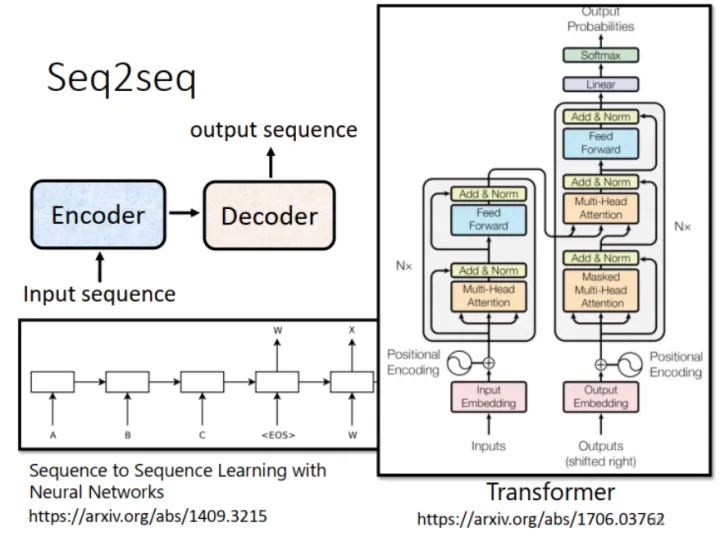
**Seq2seq for Syntactic Parsing:**如下图所示，把Syntactic Parsing（语法分析）的结果parsing tree组成sequence的形式，也可以用seq2seq做。



**Seq2seq for Multi-label Classification:**Multi-label Classification和Multi-class Classification都是输出有多个类别(class)可选，但是它们不一样。Multi-class：一个输入只输出一个类别。Multi-label：一个输入可以输出多个类别。如果你整理过如印象笔记之类的云笔记，可能有这样的体会：按类别设置了多个笔记本，比如：“学习”，“生活”，等。一条笔记只能归类放在其中的一个笔记本中（这就是Multi-class）。但是，你可以给这条笔记打上多个不同的标签 （这就是Multi-label）。Seq2seq模型由机器决定输出sequence长度。Multi-label Classification任务中输出labels个数是不确定的，因此可以应用Seq2seq模型。

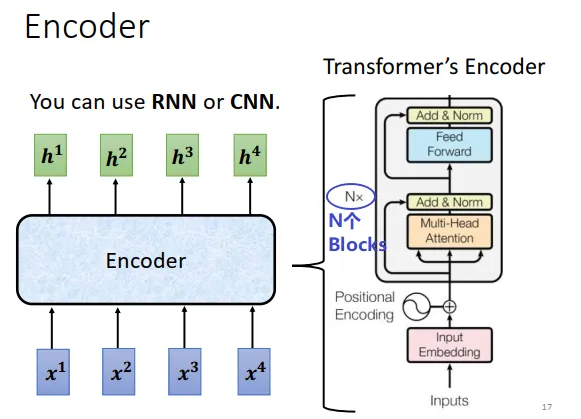


**Seq2seq 模型**

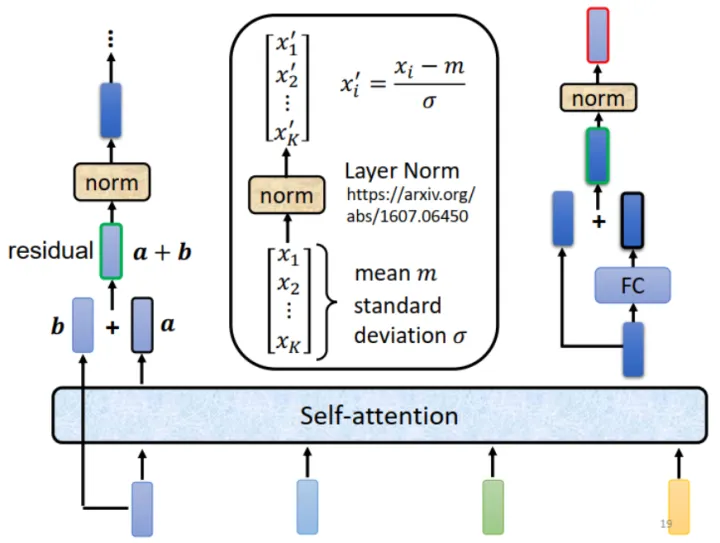


**Encoder：**

如下图所示，在Transformer的Encoder部分，有N个Blocks，每个Block都是由Self-attention（图中“Multi-Head Attention”单元）和FullyConnected Layer（图中“Feed Forward”单元）组成。



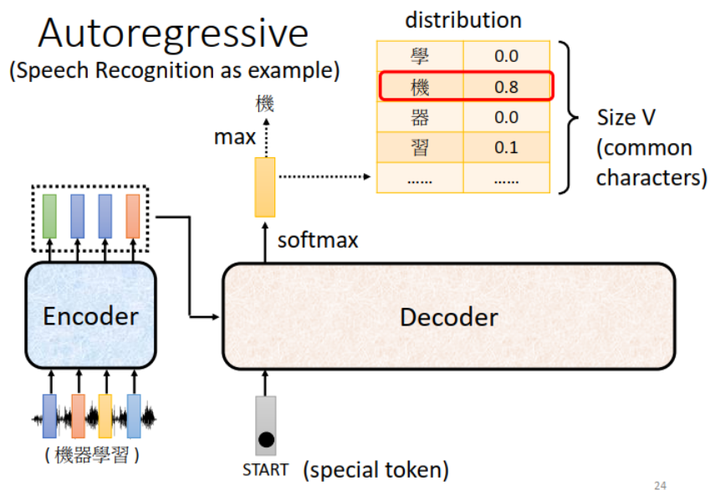
如下图所示，Add就是加上输入向量自身，这个结构叫做residual，是一个常用的网络结构设计。因为要加上输入vector，所以输入输出vector维度应保持一致，transformer论文中把每一层输出vector的维度都设为512。Norm做的是Layer Normalization，与前一课介绍的Batch Normalization不同。Batch Normalization在不同samples的相同维度上求均值方差，每个维度都有对应的均值方差。Layer Normalization在同一sample的所有维度上求均值方差，如下图所示。



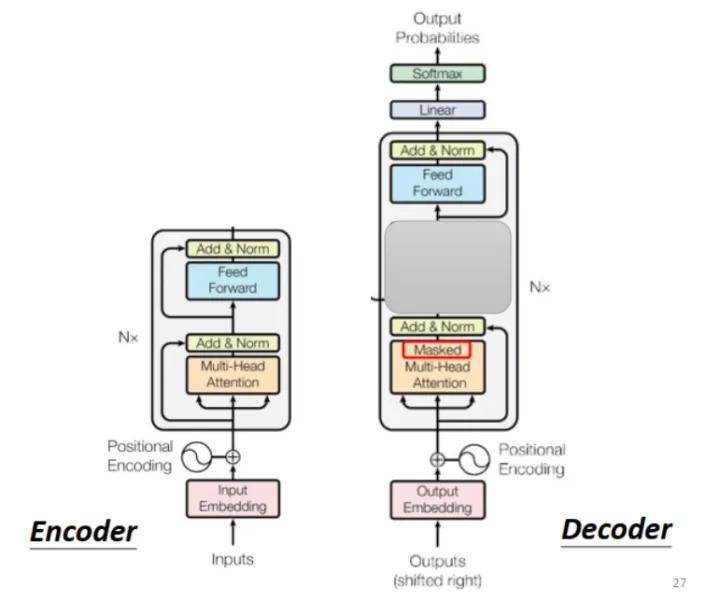
**Decoder：**

**AT (Autoregressive)：**

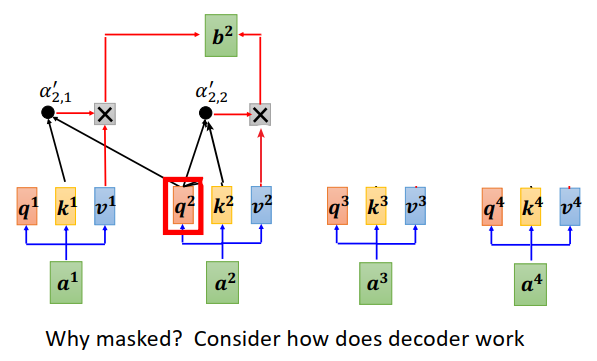
Autoregressive就是指前一时刻的输出，作为下一时刻的输入。如下图所示，起始时要输入一个特别的Token（图中的“Start”），告诉decoder：一个新的sequence开始了！这个Token经过Decoder，输出一个vector，里面是各类别的概率，从中选出最大概率对应的类别，如图示例子的“机”字，作为输出。“机”字对应的变量又作为第二个输入，经过Decoder得到第二个输出：“器”，依此类推。



再来看看decoder的内部结构。先把decoder和encoder放在一起对比看看，如下图所示，除去灰色方框遮住的部分，decoder和encoder非常相似。

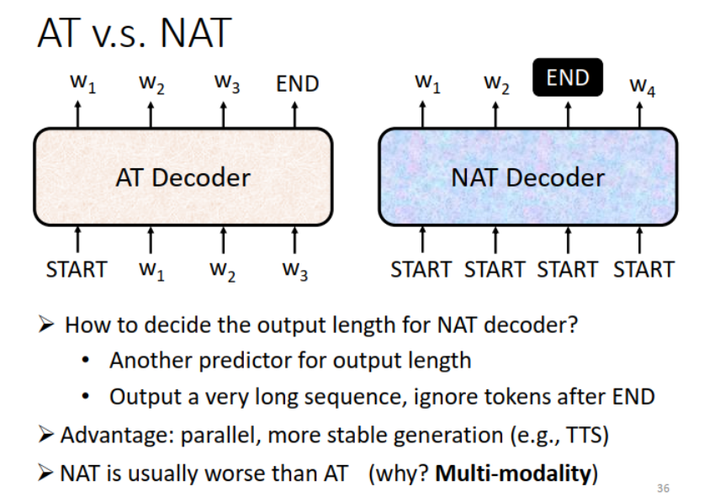


Self-attention中，每个输入vector的query会和sequence上每个vector的(key,value)做运算。



**NAT (Non-autoregressive)**

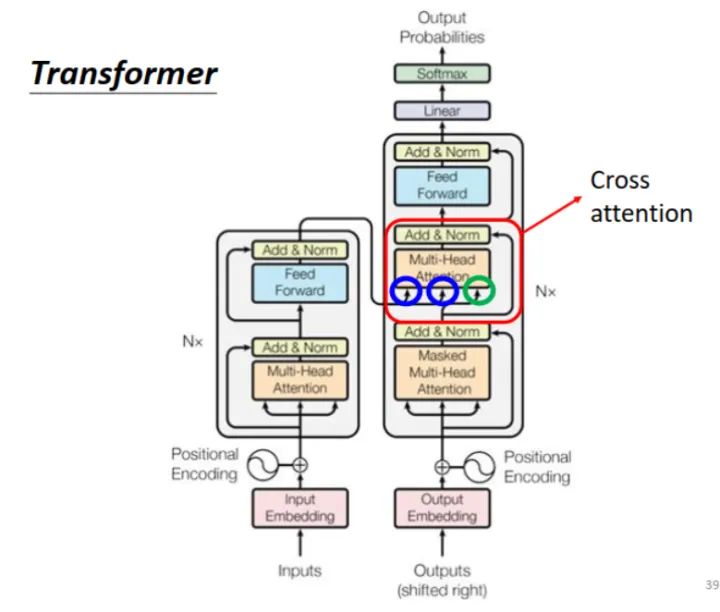
NAT 并不使用之前时刻的输出，而是一次输入一组special token。



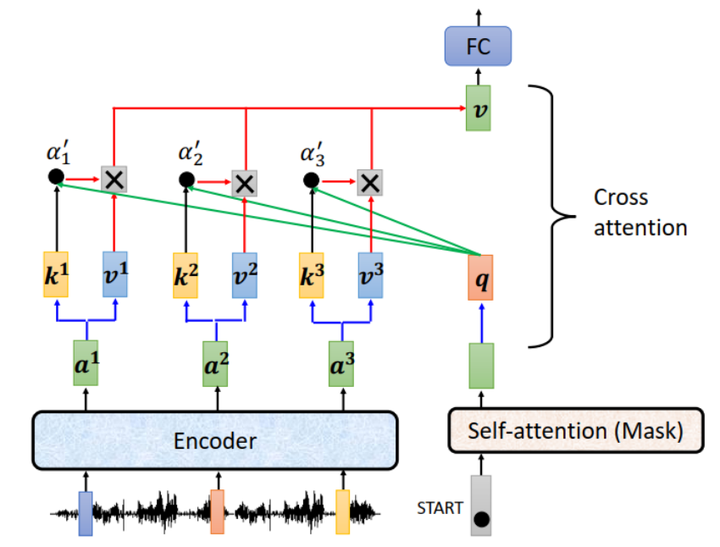
这两者各有利弊：AT一次输出一个vector（因为上一个输出又作为下一个输入），无法并行处理。而NAT可以并行处理。NAT是在transformer提出self-attention之后才出现的。为什么呢？上一节课提到，RNN也是对vector一个一个地处理，没办法做并行。self-attention才是并行处理，一次处理一个sequence。

**cross attention：**

它计算的是encoder的输出与当前vector的cross attention。

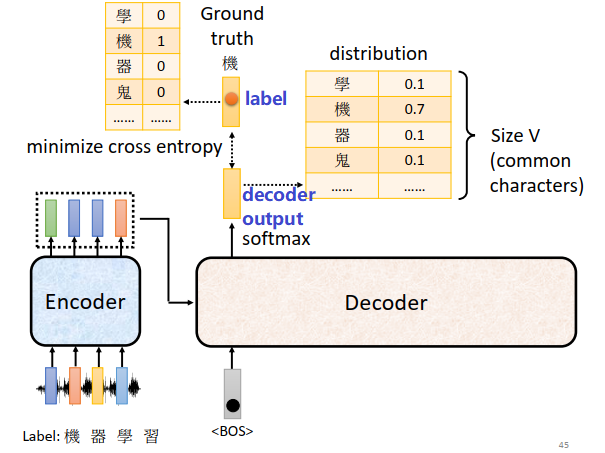


具体操作如下：



**Training Process：**

如下图所示，decoder的输出(output)是一个概率分布，label是one-hot vector，优化的目标就是使label与decoder output之间的cross entropy最小。这其实是一个分类问题。

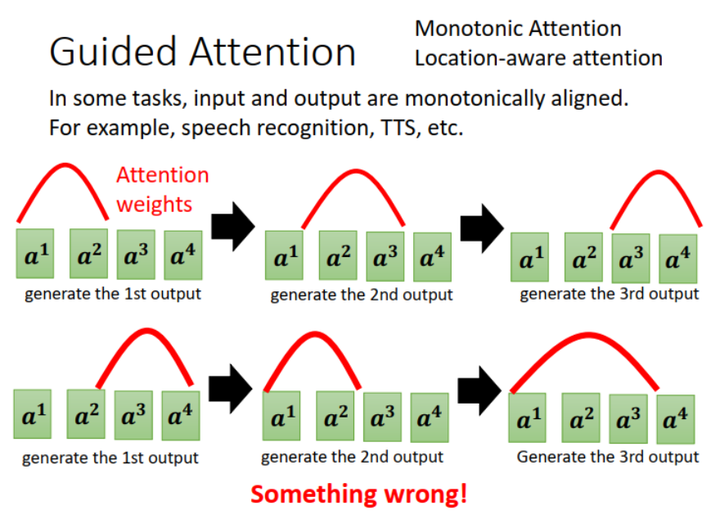


**Copy Mechnism：**

有些情况，不需要对输入做改动，比如翻译人名地名，聊天机器人(chat-bot)，摘要 (summarization) 等，可以直接复制一部分输入内容。

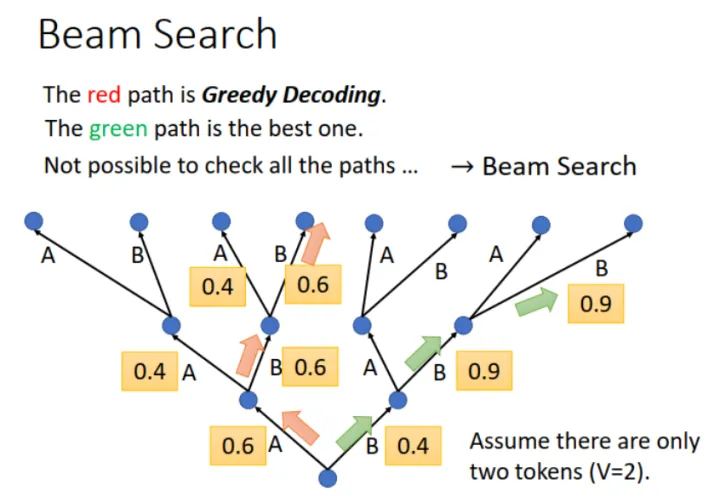
**Guided Attention：**

Guided Attention是让attention的计算按照一定顺序来进行。比如在做语音合成时，attention的计算应该从左向右推进。



**Beam Search：**

这其实是一个最优路径的问题。前面介绍，decoder每次输出一个变量，假设输出词汇库只有A,B两个词汇。每一次都选择最大概率的作为输出，如下图中红色路径所示，这就是Greedy Decoding。同时，decoder的每个输出又是下一时刻输入，如果我们从整个sequence的角度考虑，可能第一次不选最大概率，后面的输出概率（把握）都很大，整体更佳，如下图中绿色路径所示。



**exposure bias：**

训练时Decoder看的都是正确的输入值（Ground Truth）。测试时如果有一个输出有错误，可能导致后面都出错。解决办法就是让训练时decoder加入一点错误的输入，让机器“见识”错误的情况，这就是Scheduling sampling。