

Lecture 5: Transformer

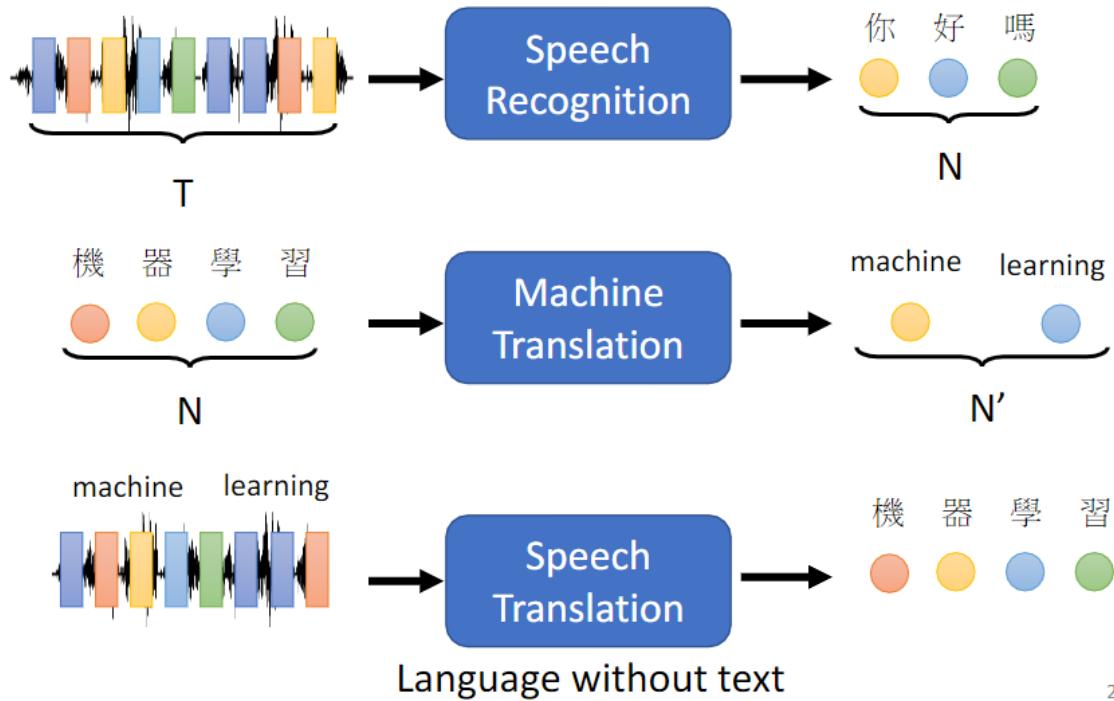
transformer和BERT很有关系

transformer就是一个**Sequence-to-sequence (Seq2seq)** 的模型。这里可以回顾下Lecture 4讲到的致力于解决输入为一组向量的深度学习任务的分类，其中模型自己决定label的数量——**seq2seq任务**

(Input a sequence, output a sequence, the output length is determined by model)

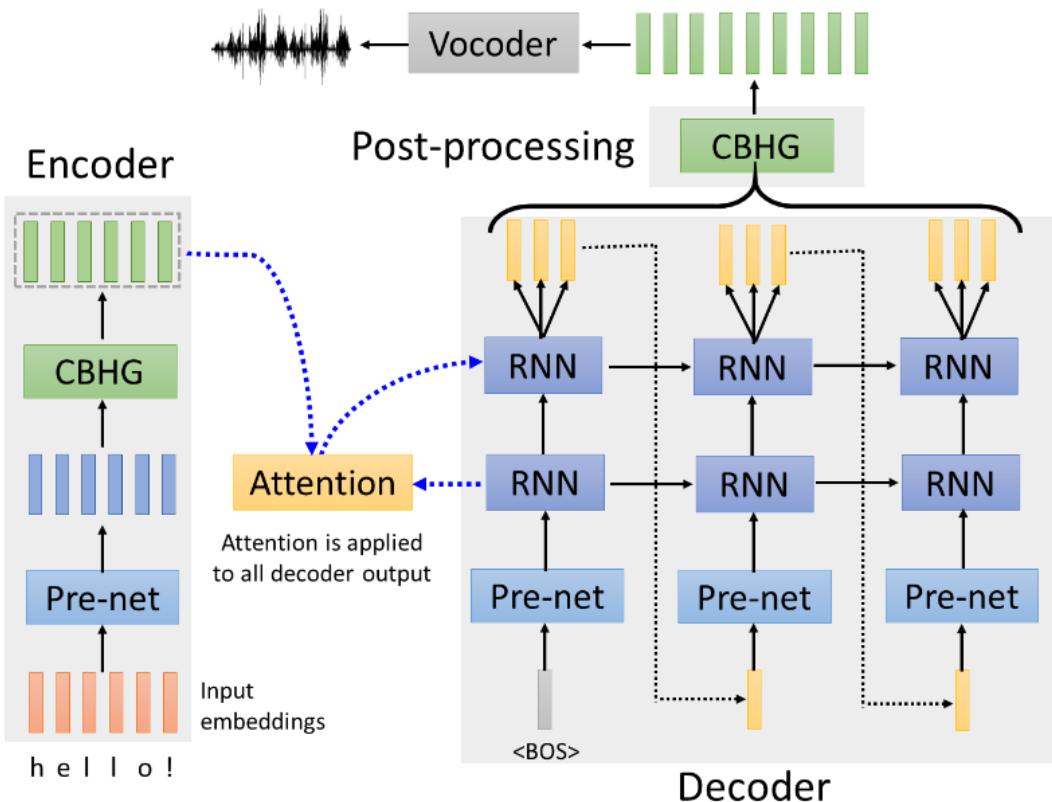
应用场景

举例来说，语音识别、机器翻译、语音翻译（世界是许多语言没有文字，所以需要直接从语音翻译）、Text-to-Speech (TTS) Synthesis (输入文字，输出语音)、seq2seq for Chatbot：聊天机器人



TTS:

2



Seq2seq for Chatbot:



Training data:

```
[PERSON 1:] Hi
[PERSON 2:] Hello ! How are you today ?
[PERSON 1:] I am good thank you , how are you.
[PERSON 2:] Great, thanks ! My children and I were just about to watch Game of Tl
[PERSON 1:] Nice ! How old are your children?
[PERSON 2:] I have four that range in age from 10 to 21. You?
[PERSON 1:] I do not have children at the moment.
[PERSON 2:] That just means you get to keep all the popcorn for yourself.
[PERSON 1:] And Cheetos at the moment!
[PERSON 2:] Good choice. Do you watch Game of Thrones?
[PERSON 1:] No, I do not have much time for TV.
[PERSON 2:] I usually spend my time painting: but, I love the show.
```

其他NLP应用场景:

Question Answering (QA)

Question	Context	Answer
What is a major importance of Southern California in relation to California and the US?	...Southern California is a major economic center for the state of California and the US....	major economic center
What is the translation from English to German?	Most of the planet is ocean water.	Der Großteil der Erde ist Meerwasser
What is the summary?	Harry Potter star Daniel Radcliffe gains access to a reported £320 million fortune...	Harry Potter star Daniel Radcliffe gets £320M fortune...
Hypothesis: Product and geography are what make cream skimming work. Entailment , neutral, or contradiction?	Premise: Conceptually cream skimming has two basic dimensions – product and geography.	Entailment
Is this sentence positive or negative? (sentiment analysis)	A stirring, funny and finally transporting re-imagining of Beauty and the Beast and 1930s horror film.	positive



QA can be done by **seq2seq**

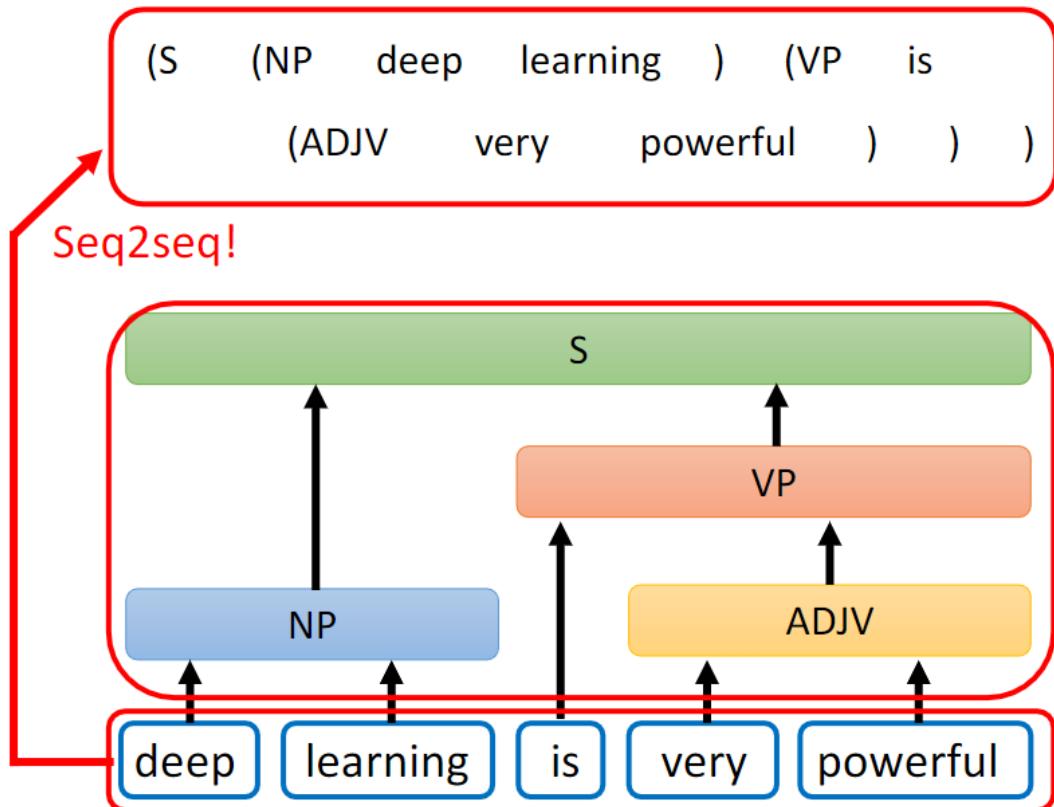


问题，文章==>答案。参考实例：[文章1](#)、[文章2](#)

强调一下：对于多数的NLP问题或对多数的语音相关的任务而言，往往为这些任务**定制化模型**，会得到更好的结果。举例来说，Google的pixel4用的N to N的network用了一个RNN transducer的model来完成语音识别任务，模型设计针对了语音的特性，那么对于相关语音任务表现会更好。

Seq2seq for Syntactic Parsing

文法剖析任务，输入一段sequence，输出一个文法剖析树



一篇文章[grammar as a foreign language](#)介绍了这个做法。用原来只用在翻译的Seq2seq的model来做文法剖析（故名为文法当作另一门“语言”，所以是硬用seq2seq）

Seq2seq for Multi-label Classification

Multi-label Classification: 同一个东西可以属于多个class。

由于每个东西可能所属类的数量种类有所不同，所以如果只是设置一个threshold (比方说取前三名) 效果可能非常不好。然而这里可以用seq2seq硬做，机器自己决定输出几个class。

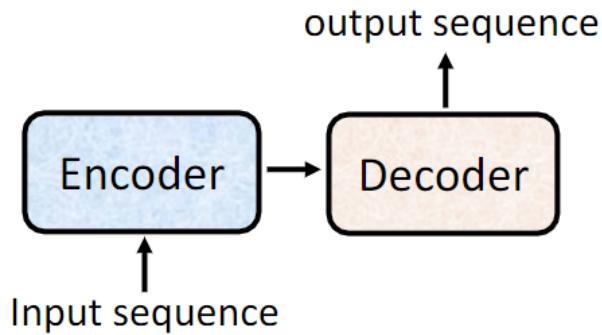
参考实例：[文章1](#)、[文章2](#)

Seq2seq for Object Detection

[参考文章](#)

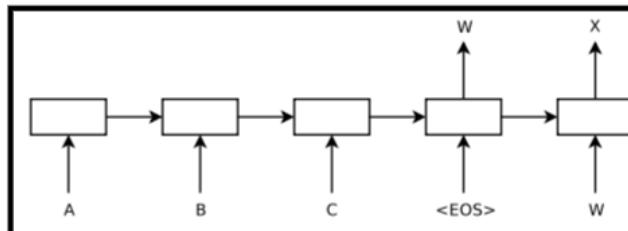
综上 seq2seq很powerful

Seq2seq的具体实现



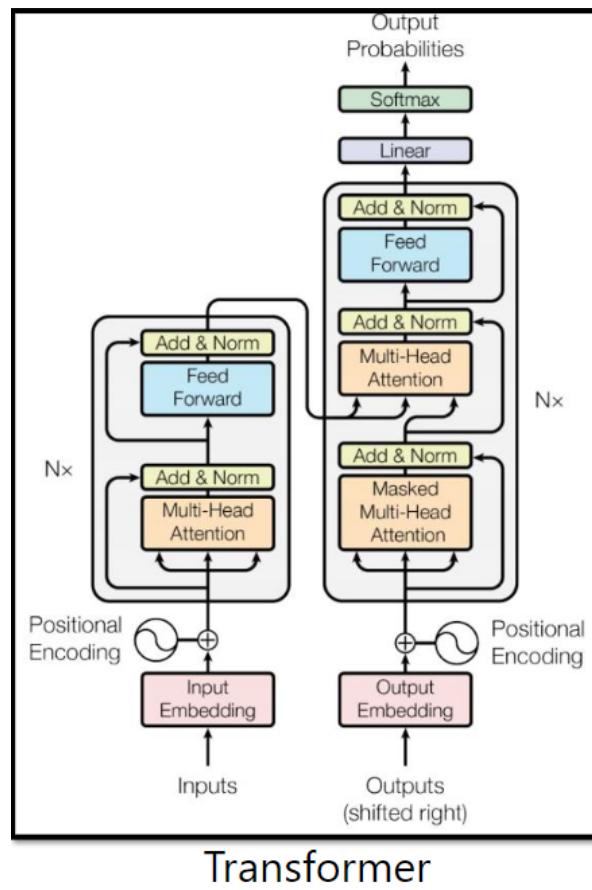
如上图，把输入sequence丢进Encoder，然后再扔进Decoder，由Decoder决定输出什么样的sequence。之后会细讲Encoder和Decoder的内部架构。

经典的[seq2seq](#)架构



Sequence to Sequence Learning with
Neural Networks

这节的主角[Transformer](#)



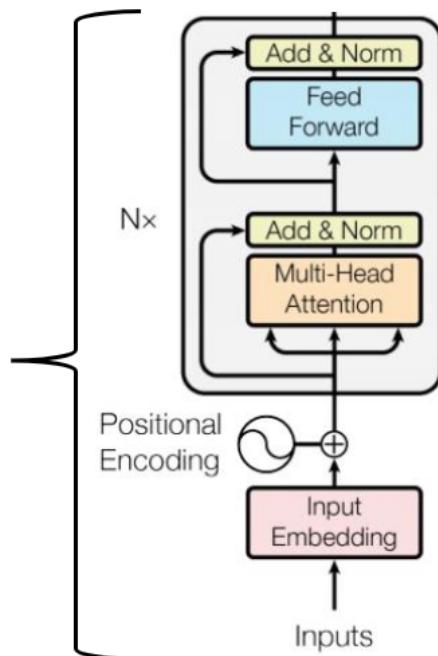
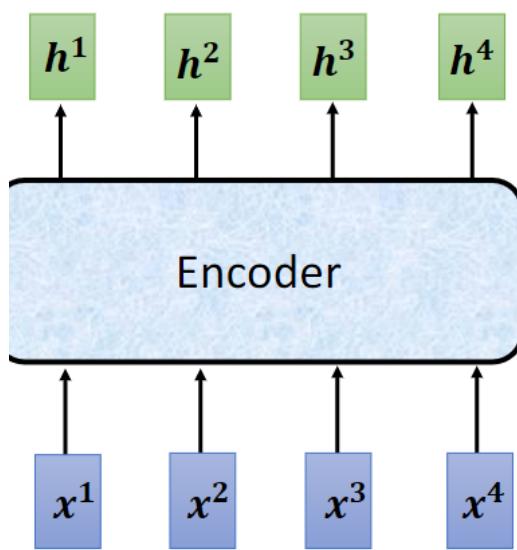
Transformer

Encoder

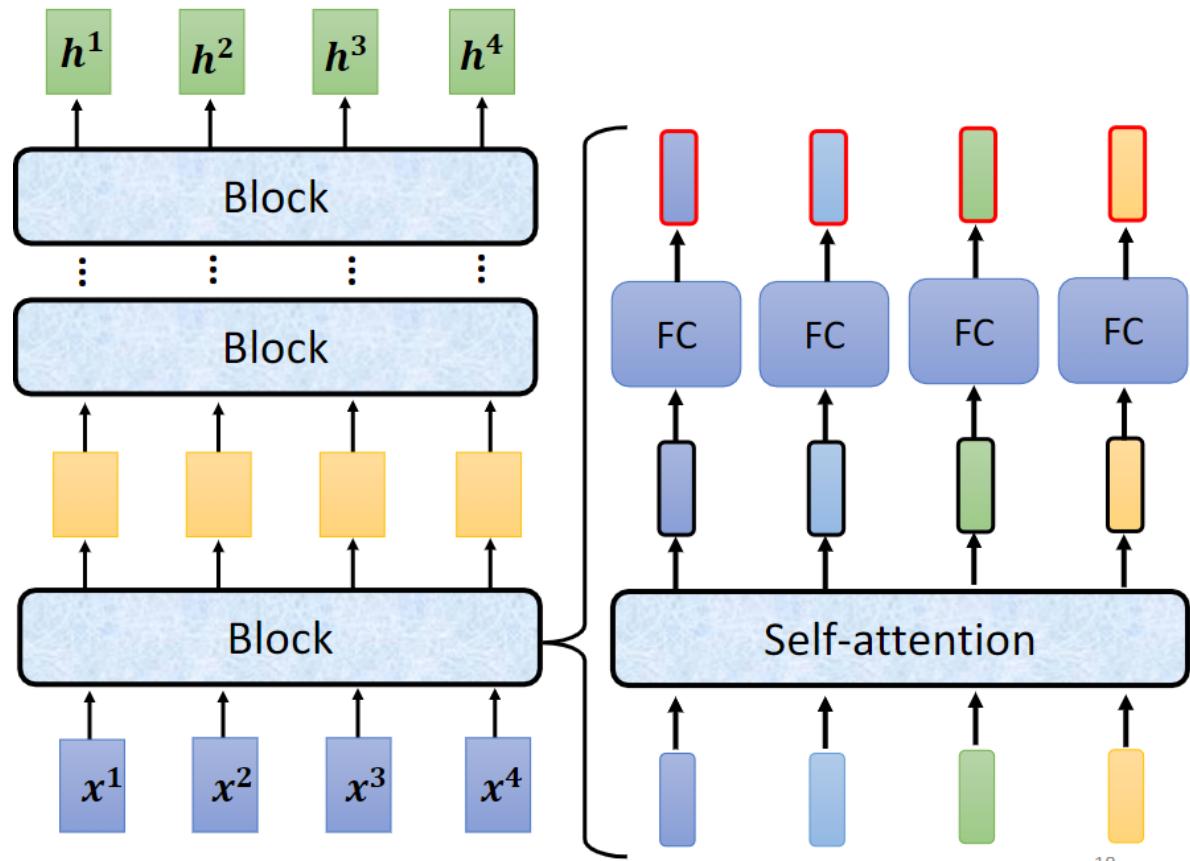
作用：input一排向量输出另外一排向量。上一个lecture讲的self-attention以及之前的RNN和CNN都能做到，而在Transformer里面的Encoder用的是self-attention。

Transformer's Encoder

You can use RNN or CNN.

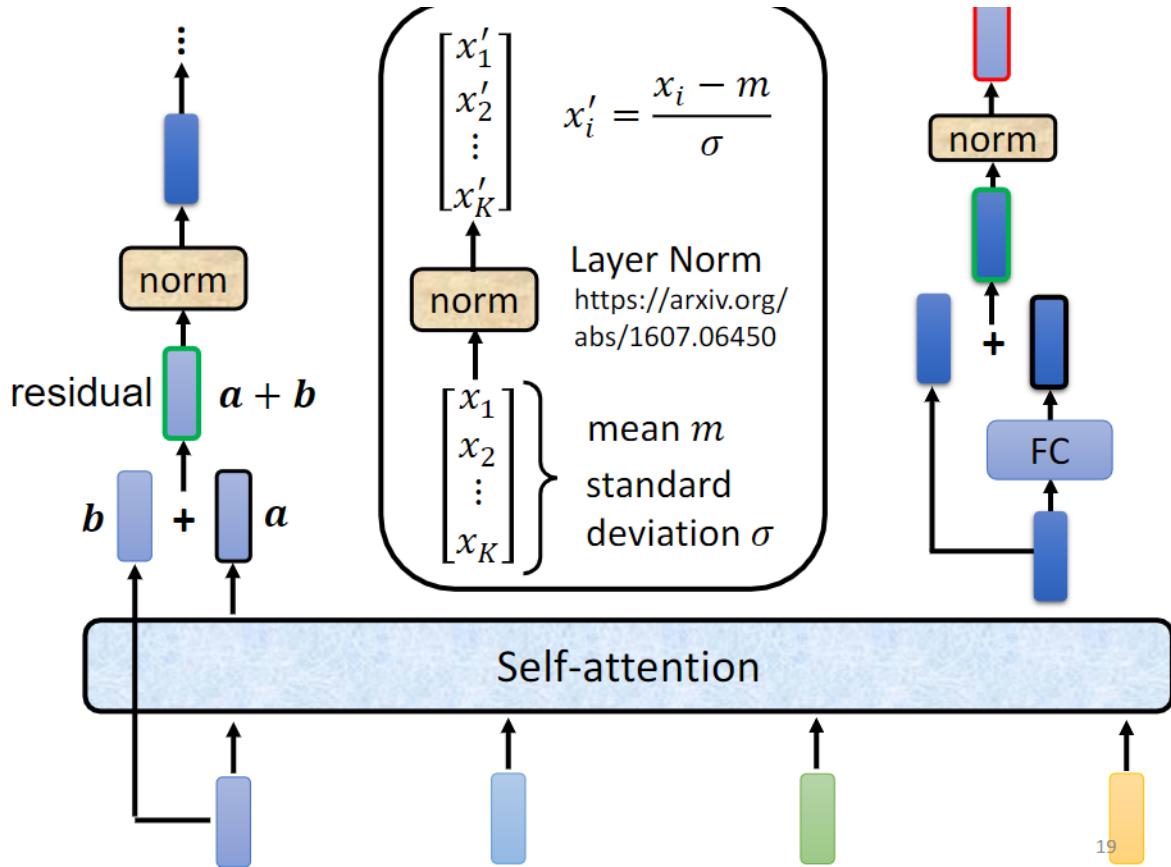


把Encoder内部肢解出来，在Transformer里面

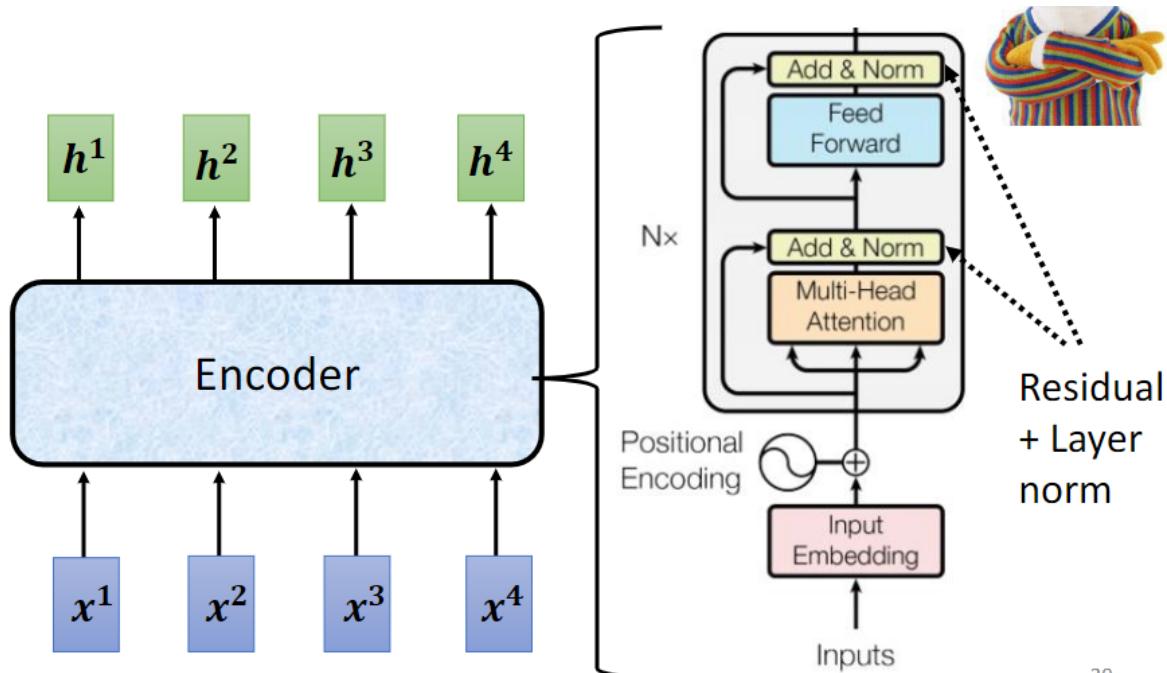


Encoder的关键是许多个Block，每个Block有好几个Layer在工作，每个Block整合了一系列的操作（self-attention + FC）：首先是输入进self-attention以考虑到整个sequence的资讯，output另外一排vector；然后把这排vector再丢进fully connected的feed forward network里面，再output另外一排vector——即Block的输出。

在原来的Transformer里面，做的事情更复杂：



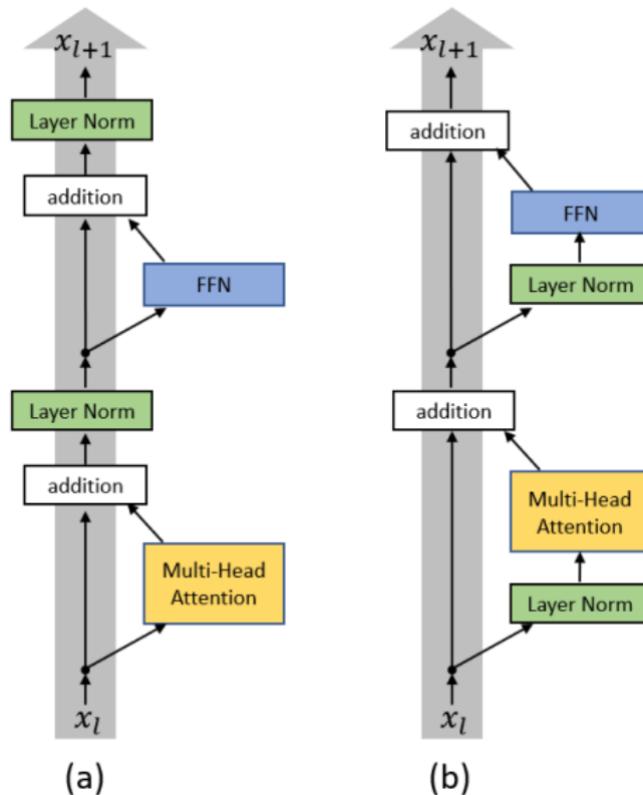
在Transformer里面，在输出的vector还要加上对应的input，包含这种形式的架构称之为**Residual connection**；接着把结果做**Layer normalization**（相较于batch normalization更简单些）。这个输出才是FC的输入，然后再做一次residual connection和layer normalization。



上图中的**Add & Norm**就是**residual connection加上layer normalization**；Positional Encoding和Multi-Head Attention之前讲过。

BERT其实就是Transformer的Encoder。

上述是原始论文的Transformer设计，相关的深入探讨：[On Layer Normalization in the Transformer Architecture](#)、[PowerNorm: Rethinking Batch Normalization in Transformers](#)



也有Transformer的魔改版本：(a)是原始的，(b)是其中一个魔改（更换了Layer Norm顺序...）

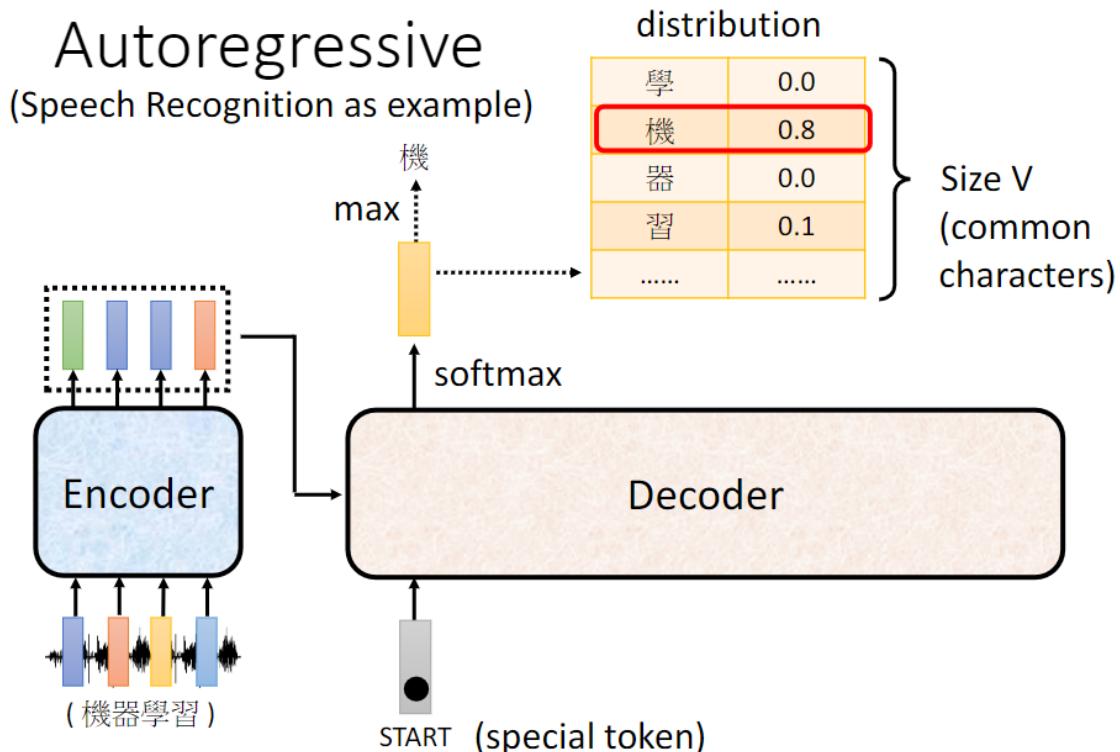
Decoder其实有两种

比较常见的一种：**Autoregressive (AT)**

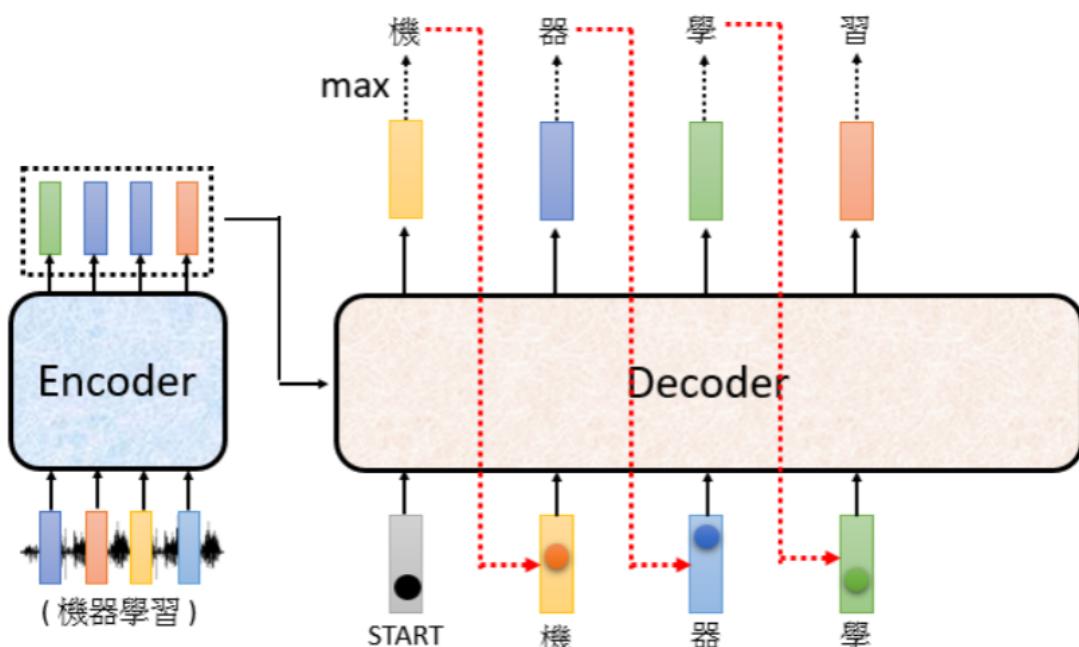
Speech Recognition (输入一段声音，输出一段文字) as example, 说明AT如何运作

Decoder读入Encoder的输出（某种方法），产生语音识别的结果。Decoder如何产生一段文字？

- 首先要先给他一个特殊的符号 START(special token)，代表start of sentence。在Decoder可能产生的一段文字开头加上一个特殊的符号（字），就代表了开始这个事情。每一个 token 都可以把当作一个 one-hot 的vector（一维是1其他是0），Decoder吐出一个长度很长的向量（和 Vocabulary的大小一样【取决于预计输出的文字形式】）可以用subword代表英语（如果用词汇那需要的维数太多了），用方块汉字代表中文（常用的不过三四千）。

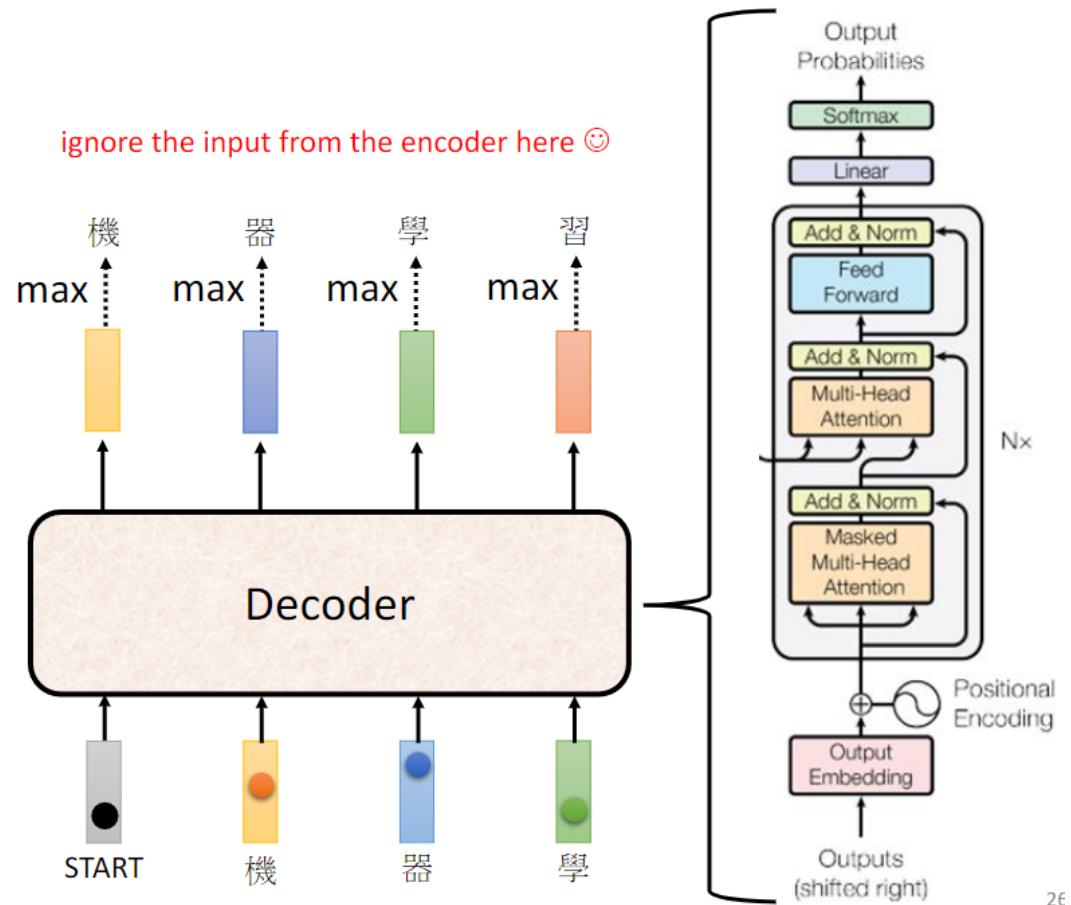


由于Decoder的初始输出要经过 softmax 处理，所以这个向量里面的值是一个distribution，向量的值加起来总和为1，选择分数最高的输出。然后把这个输出当作第二个token输入，以此类推：

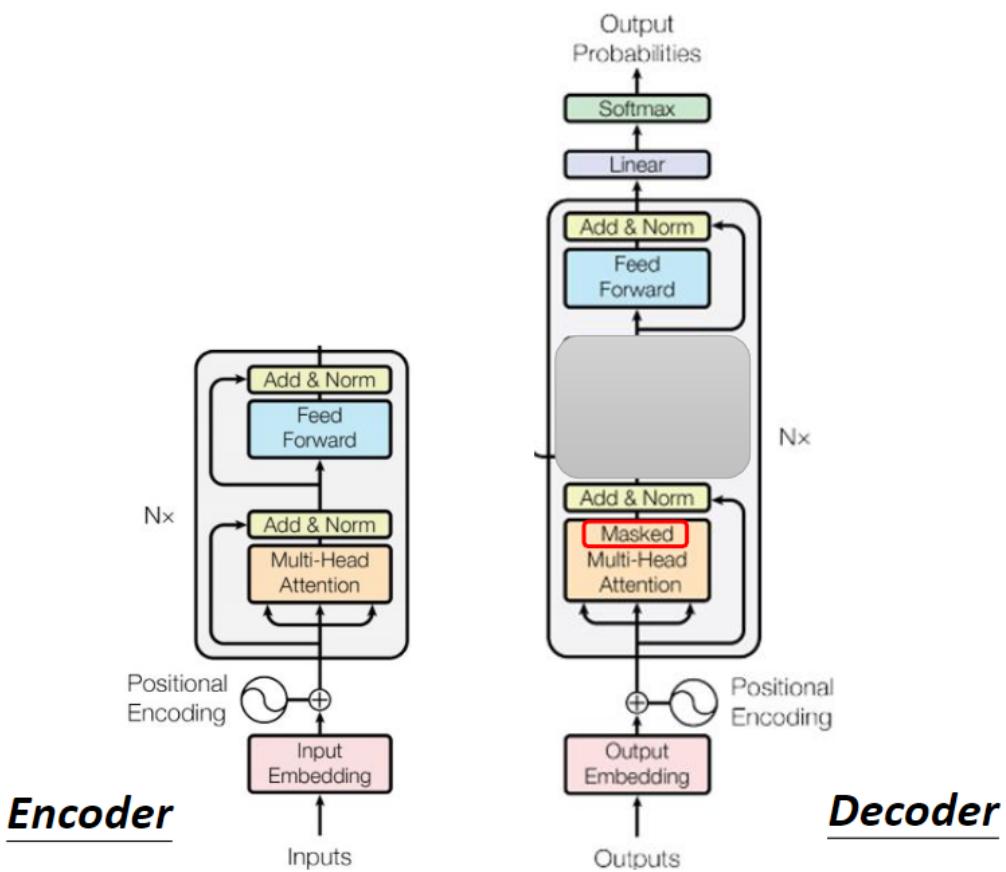


把前面的输出当作下一个输入，再参照之前的输入得到相应的输出。因为它只能看到的是自己的输出，如果其中一步有错误，而错误的输出被输入进Decoder，所以可能会产生一连贯错误。**Error Propagation**（一步错，步步错）。这个问题之后再讨论。

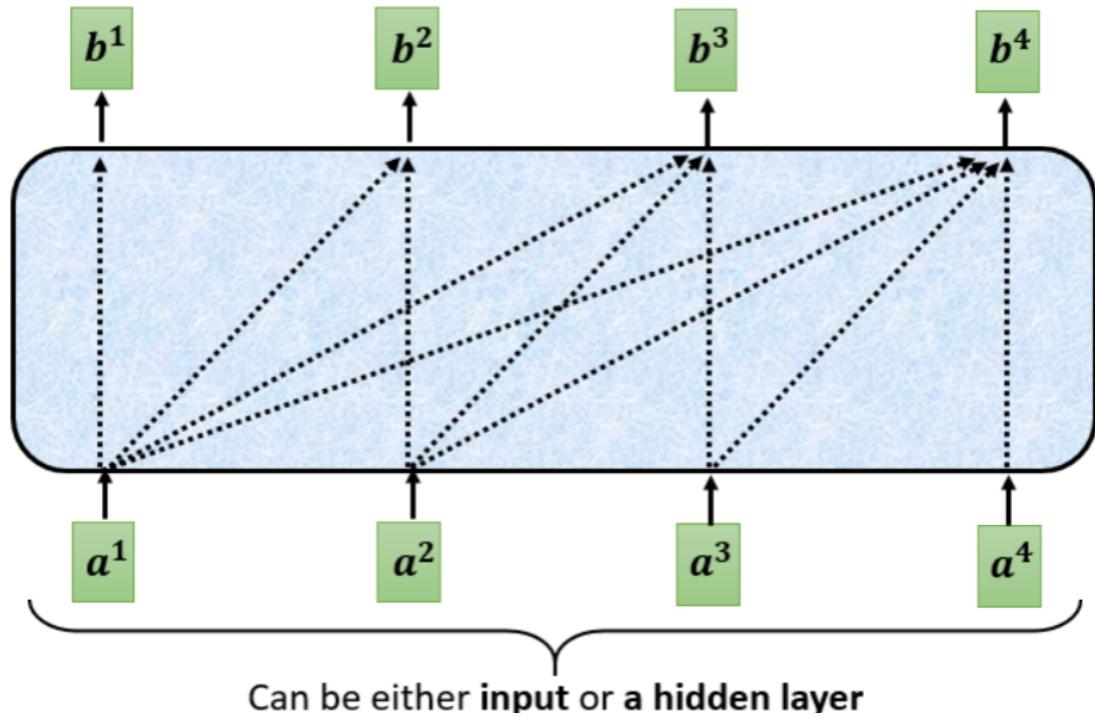
现在把Encoder的部分省略掉，留下Decoder，**Decoder的内部结构长什么样？**（以Transformer为例，比Encoder复杂多了😊）



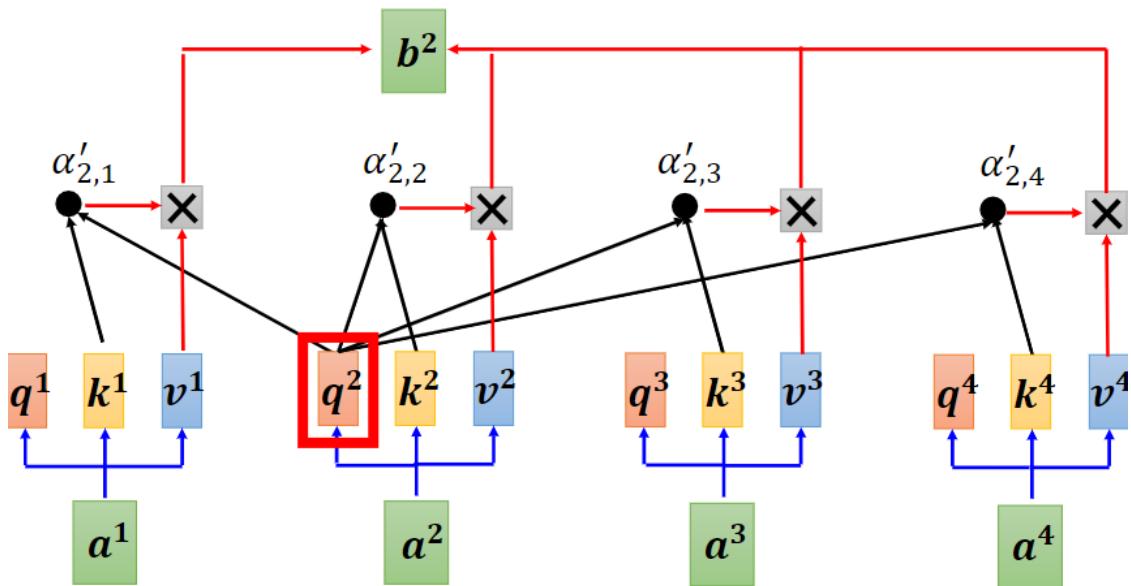
除了中间一块马赛克部分，其实Encoder和Decoder差别不是很大😊



在Decoder这边输出多了 softmax 部分，而在Blocks里面不一样的是**Masked Multi-Head Attention**；回顾一下**self-attention**：每个output都会考虑所有的input；而**Masked Self-attention**：每个output只考虑相对之后的input。



如图所示：



为什么要masked呢？原因很直觉——它的输入是一个一个打进去的（而非并行输入）。

有一个关键的问题是——Decoder必须自己决定输出的sequence的长度。那么Decoder如何决定什么时候输出sequence的截止？答案是Decoder需要设计一个特别的符号：断 (END) 来表示，这个就是“**Stop Token**”。

distribution

學	0.0
機	0.8
器	0.0
習	0.1
.....
END	0.0

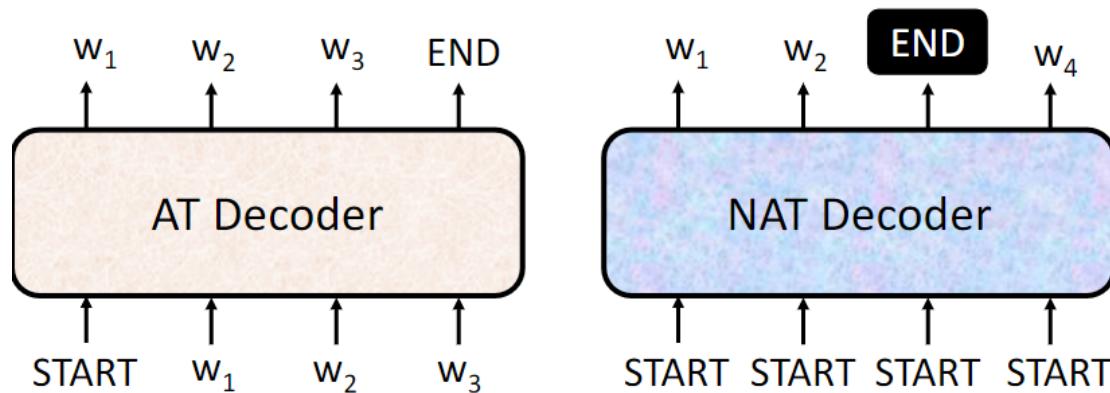
} Size V
 (common
 characters)

有始必有终 (Start—End)，程序中也可以两者作为同一种符号，只不过一个sequence开始输出，另一个sequence结束输出，也是没有问题的。整个流程的描述：Decoder看到了Encoder的embedding以及看到了Begin和自己之后的滞后的输入，就知道识别结束，输出一个END符号（即END符号的机率必须最大）。

Decoder—Non-autoregressive (NAT)

这个model以下简单介绍下。

AT v.s. NAT



NAT不像AT，一个一个输出，而是把整个句子都产生出来。NAT的Decoder吃进去一排START的Token，只要一个步骤就可以完成句子的生成。

- 那么，如何决定NAT decoder的输出长度呢？

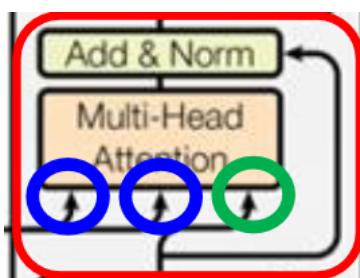
- 1、另外做一个predictor，来预测输出长度：可以是一个classifier，把decoder的输出扔进去，然后预测出一个数字——决定了有几个start token打进去。
- 2、Output a very long sequence, ignore tokens after END；不管三七二十一，输出足够多的start token，在冗余的输出中找到END特殊字符 (stop token) 截取左边作为有效输出。结束。

- NAT的decoder有什么样的好处呢？

- 1、并行化 (AT是串行的，所以NATdecoder速度更快) ’
 - 2、相较于AT更能够控制它输出的长度：因为由上我们经常使用一个classifier来决定输出长度，在一些任务中（例如语音识别），可以classifier的output除以2，速度就变快2倍。
- NAT通常表现不如AT (Why? **Multi-modality**)

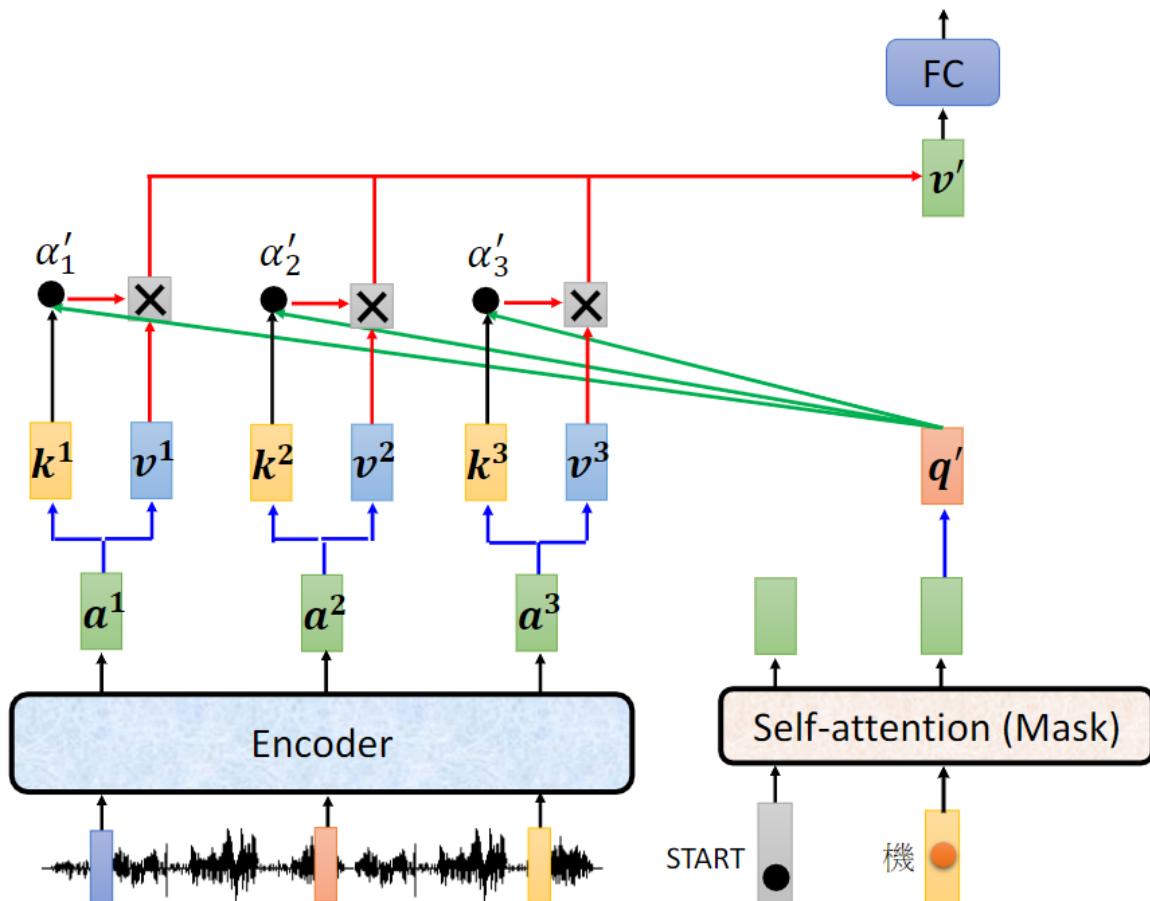
Encoder和Decoder是如何传输资讯的呢？

方才比较Encoder和Decoder的结构时，Decoder被马赛克的一块

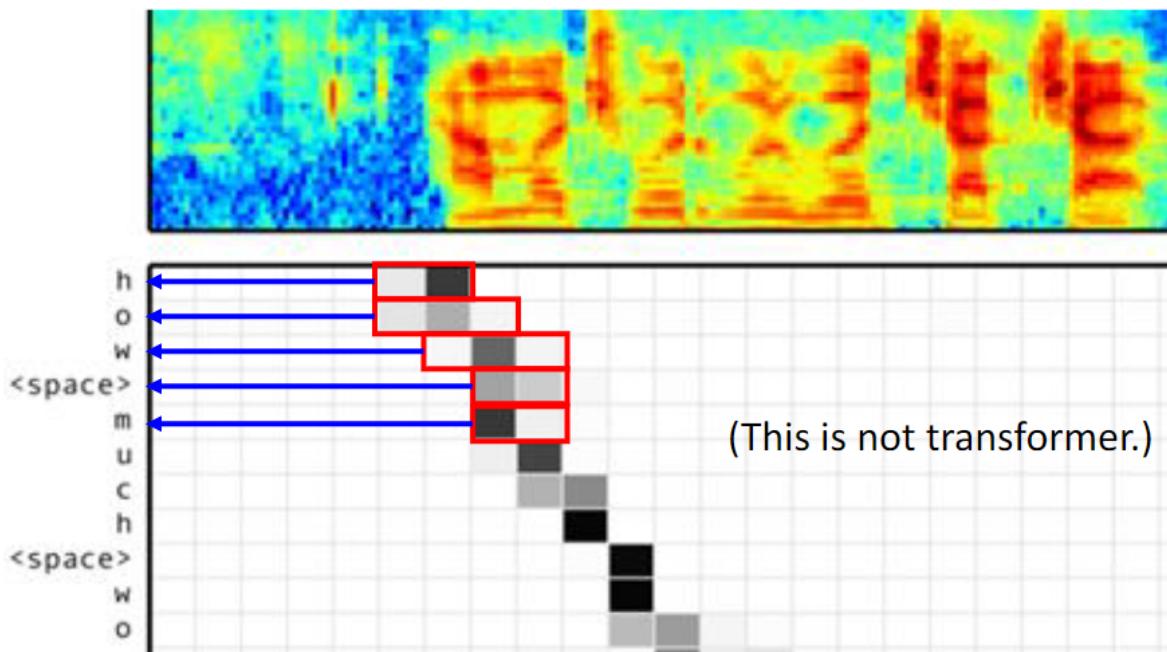


这个结构称之为**Cross Attention**。这个模组实际上是如下运作的：

Decoder会先读进start的一个token，其中的self-attention是mask的，得到一个向量，做transform，乘以一个矩阵 W_q 得到一个 q 。同时Encoder会依照self-attention产生 a^i, k^i, v^i 由此计算得到 α_i 。这个步骤称之为**Cross Attention**。（ q 来自Decoder， a, k, v 来自Encoder）之后的计算和self-attention类似，得到加权值在输入全连接层中。



来自ICASS的文章[Listen, attend and spell: A neural network for large vocabulary conversational speech recognition](#)——seq2seq硬做语音识别，稍逊于SOTA。有趣的是，这里已经出现attention的机制，是先有cross attention再有self-attention。



attention分数较高的部分几乎就是从左往右偏移（mode至上而下）。

原论文中会用Encoder最后一层输出作为最终输出，也可以不这样，多一些另外新的想法，也有人尝试不同的cross-attention，例如[文章](#)。这里也可以是一个研究问题。

语音识别任务在实际的train下，我们希望distribution的概率值要和ground truth（通常就是one-hot vector）越接近越好（两者做cross entropy得到的值越小越好），这玩意和分类很像，每次在通过一段语音产生一个中文字的过程都像是一个分类过程。每次“分类”都会计算一次cross entropy，每次总和起来的entropy期望其越来越小。注意到还有END和START的特殊字符，那么这个位置的distribution值应当与对应特殊字符的one-hot的cross-entropy越小越好。另外，在训练的时候Decoder会输入ground truth而非我们之前所提到的前一个distribution——这件事叫做**Teacher Forcing**: using the ground truth as input。

但是test的时候，显然不能把正确答案input进去，这时候decoder只能看到自己的输入和输出，这中间显然有一个Mismatch。

这两者的mismatch的现象称之为**exposure bias**，怎么解决（缓解）呢？直接在Decoder的输入加一点noise，它反而会学的更好。

- Scheduled Sampling

参考文献列表：[Original Scheduled Sampling](#)、[Scheduled Sampling for Transformer](#)、[Parallel Scheduled Sampling](#)

使用Scheduled Sampling会伤害到Transformer的平行化。所以transformer的scheduled sampling招数和传统的不一样。

训练seq2seq模型的tips

Copy Mechanism

让Decoder从输入“复制”一些东西进来当作输出，比如聊天机器人

Chat-bot

User: X寶你好，我是庫洛洛

Machine: 庫洛洛你好，很高興認識你

还有比如说做摘要(Summarization)任务，[参考文献](#)

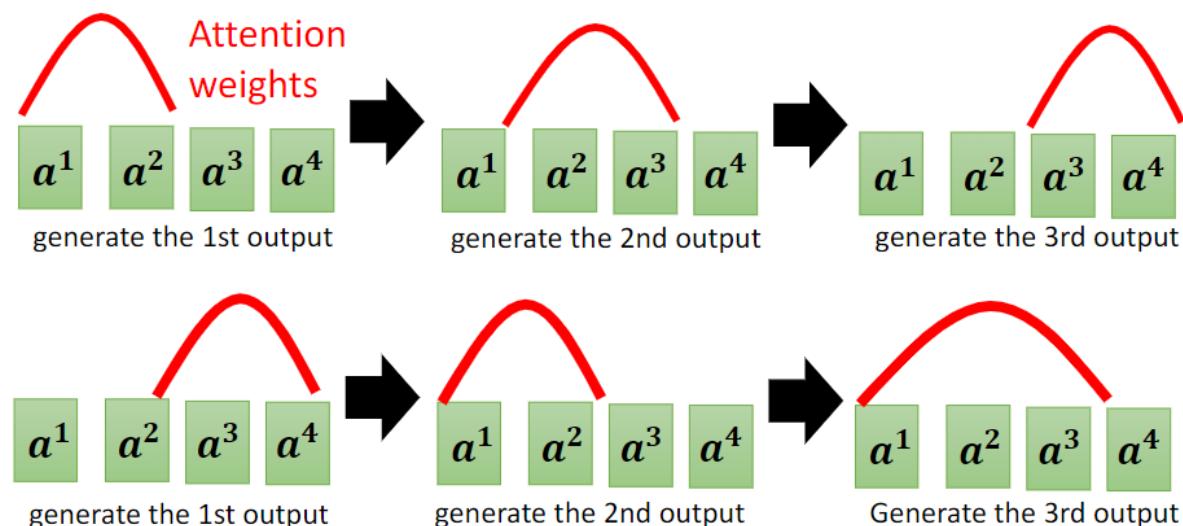
最早从输入复制东西当输出的一部分的模型叫做Pointer Network,后来有一篇变形[Copy Network](#)

Guided Attention

要求机器去领导attention的过程，使其有固定的方式进行。

In some tasks, input and output are monotonically aligned.

For example, speech recognition, TTS, etc.



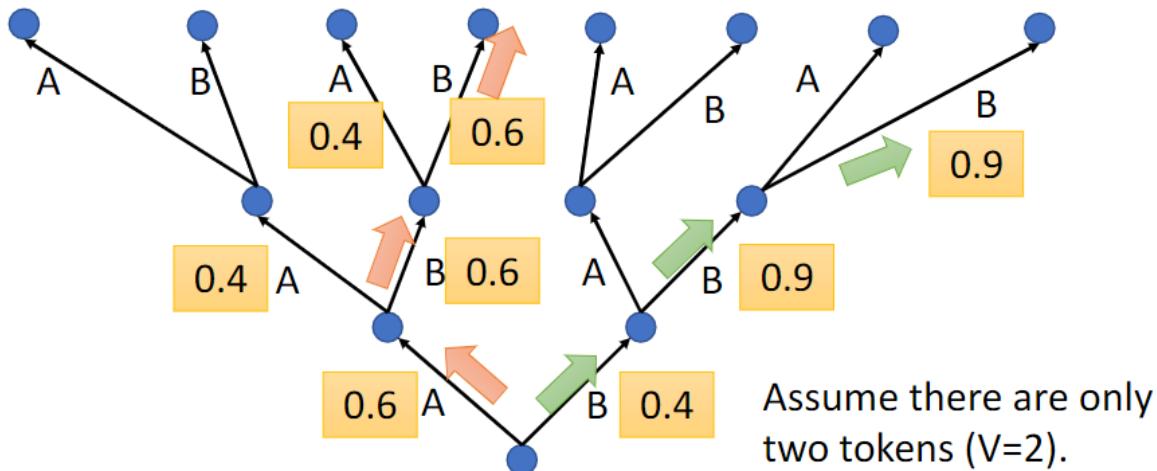
- Monotonic Attention
- Location-aware Attention

Beam Search: 集束搜索

The **red** path is **Greedy Decoding**.

The **green** path is the best one.

Not possible to check all the paths ... → Beam Search



- Greedy Decoding

对于目的和答案比较明确的任务，Beam Search比较有帮助。但是对于一些需要机器发挥“创造力”的任务，不太好用。在实践中，加入一些随机性（noise）可能结果会出乎意料的好。

遇到用optimization解决不了的问题，用RL（Reinforcement Learning）硬train一发就对了。把loss function当作是award，把你的decoder当作是Agent，是有可能可以做的。[\[1511.06732\] Sequence Level Training with Recurrent Neural Networks \(arxiv.org\)](#)