05-Transformer

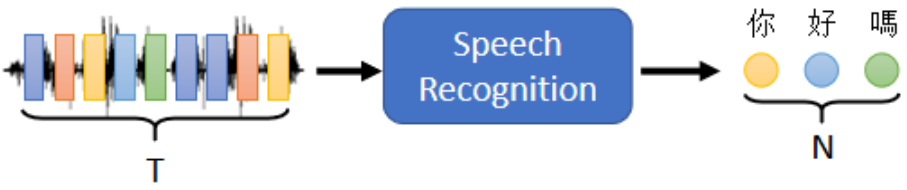
1. Seq2Seq 模型

Transformer 是一個序列到序列（Sequence-to-Sequence，Seq2Seq）的模型。序列到序列模型**輸入和輸出都是一個序列**，輸入與輸出序列長度之間的關係有兩種情況：一是**輸入跟輸出的長度一樣**；二是**機器自行決定輸出的長度**。

2. 應用

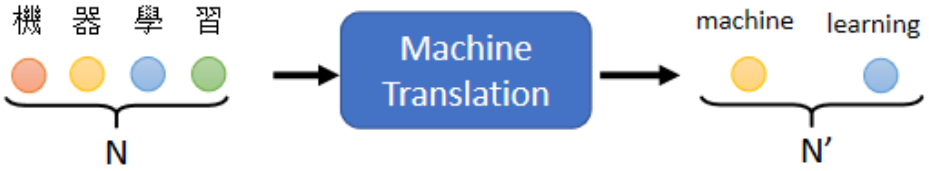
2.1 語音識別

輸入是聲音訊號的一串的 vector，輸出是語音辨識的結果，也就是輸出的這段聲音訊號，所對應的文字 ⇒ **輸出的長度由機器自己決定**



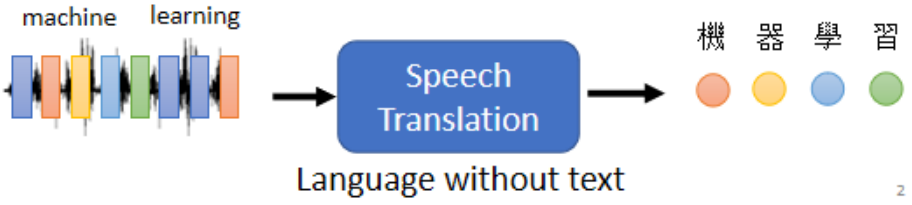
2.2 機器翻譯

機器讀一個語言的句子，輸出另外一個語言的句子，輸入的文字的長度是 N，輸出的句子的長度是 N'，N 跟 N' 之間的關係也**由機器自己來決定**



2.3 語音翻譯

將聽到的英文的聲音訊號翻譯成中文文字



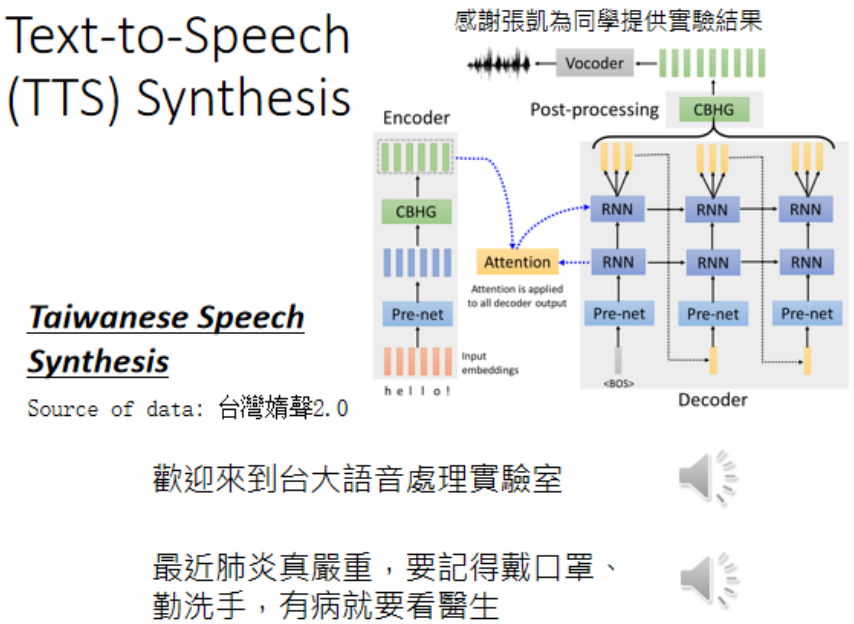
問題：

把語音識別系統跟機器翻譯系統接起來就直接是語音翻譯，為何還需獨立出語音翻譯？
因為世界上很多語言沒有文字，無法做語音識別。因此需要對這些語言做語音翻譯，直接把它翻譯成文字

2.4 語音合成

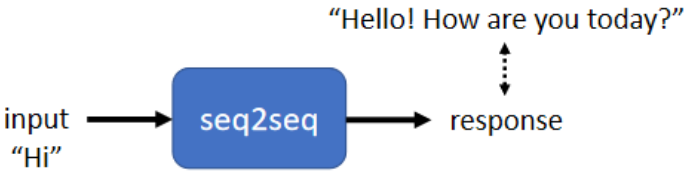
輸入文字、輸出聲音信號就是**語音合成（Text-To-Speech，TTS）**

現在還沒有真的做端到端（end-to-end）的模型，以閩南語的語音合成為例，其使用的模型還是分成兩階，首先模型會先把中文的文字轉成閩南語的拼音，再把閩南語的拼音轉成聲音信號



2.5 聊天機器人

因為聊天機器人的輸入輸出都是文字，文字是一個向量序列，所以可用序列到序列的模型來做一個聊天機器人



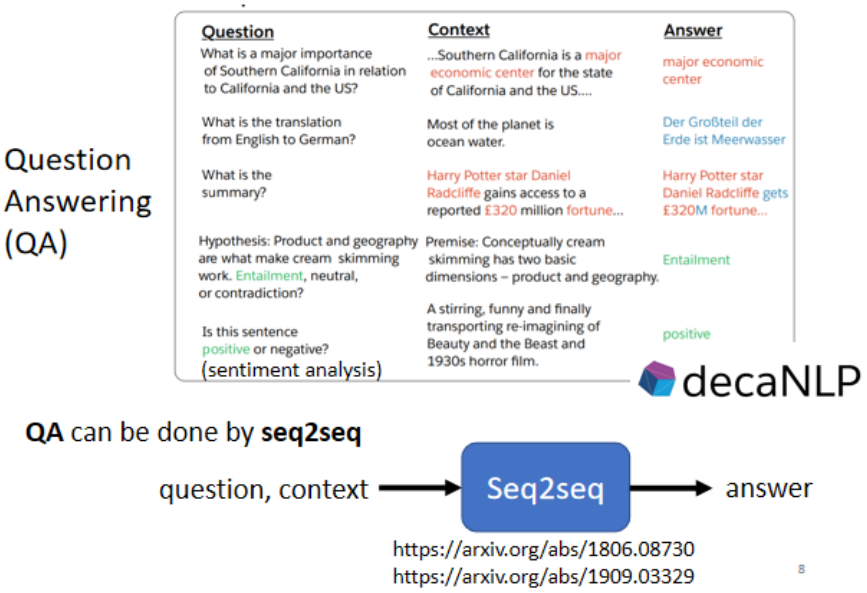
Training data:

[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 Thrones.
[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.

2.6 QA 任務

很多自然語言處理的任務都可以想成是**問答（Question Answering，QA）**的任務，如：

- 翻譯
- 自動摘要
- 情感分析

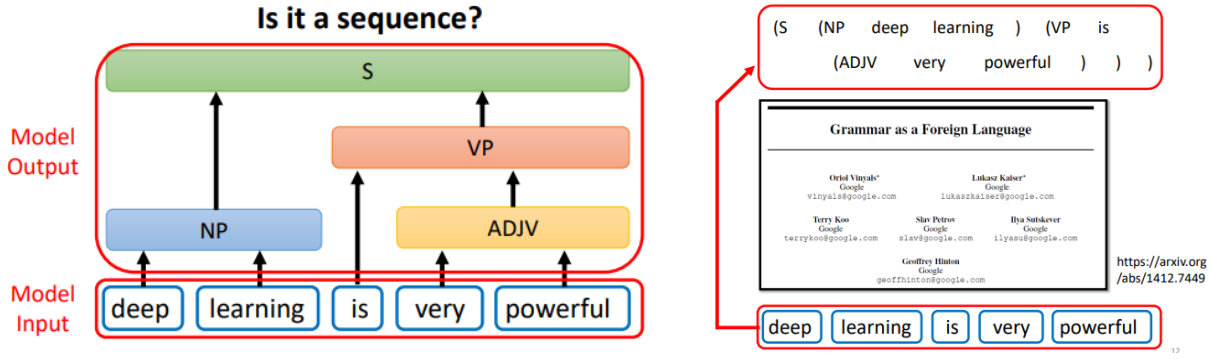


各種 NLP 問題都能用 Seq2seq 模型來解，但對多數 NLP 任務而言，**為各種不同的任務客制化模型往往比只用 Seq2seq 模型的模型更好**。例如 Pixel 4 手機用於語音識別的模型不是 Seq2seq 模型，而是 RNN-Transducer 模型，這種模型是為了語音的某些特性所設計的表現更好

學習更多：https://speech.ee.ntu.edu.tw/~hylee/dlhlp/2020-spring.html

2.7 文法剖析

文法剖析任務中，**輸入是一段文字，輸出是一個樹狀的結構**，而一個樹狀的結構可以看成一個序列，該序列代表了這個樹的結構



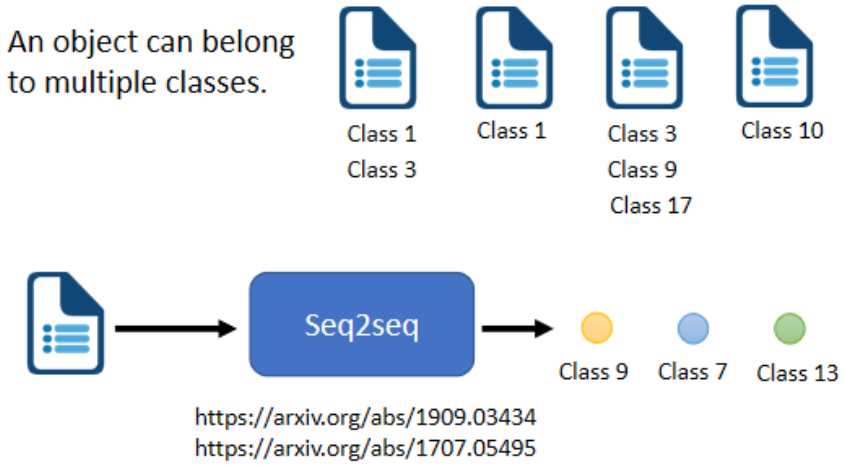
把樹的結構轉成一個序列以後，就可以用 Seq2seq 模型來做文法剖析，具體可參考論文：*Grammar as a Foreign Language*

2.8 多標籤（Multi-label）分類

區分：

- Multi-class：從數個 class 裡面選擇某一個 class

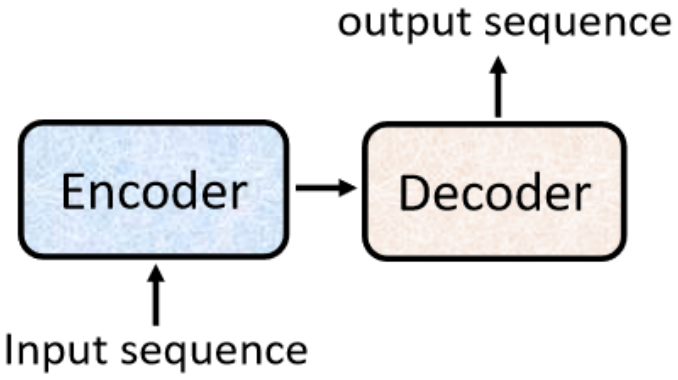
- Multi-label：同一個 sample 可以屬於多個 class



多標簽分類（multi-label classification）問題**不能直接把它當作一個多分類問題的問題來解**。就算取一個 threshold，只輸出分數最高的前三名，但因 sample 對應的類別的數量可能根本不一樣，因此需要用 Seq2seq 模型，讓機器自己決定輸出類別的數量。
可參考論文：
End-to-End Object Detection with Transformers

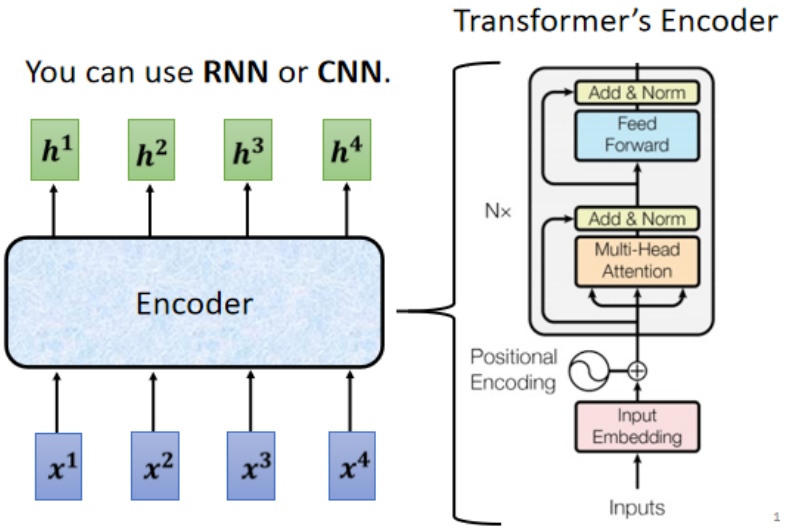
3. Transformer 架構

一般的 Seq2Seq 模型會分成 **encoder** 和 **decoder**，encoder 負責處理輸入的序列，再把處理好的結果給 decoder 決定要輸出的序列



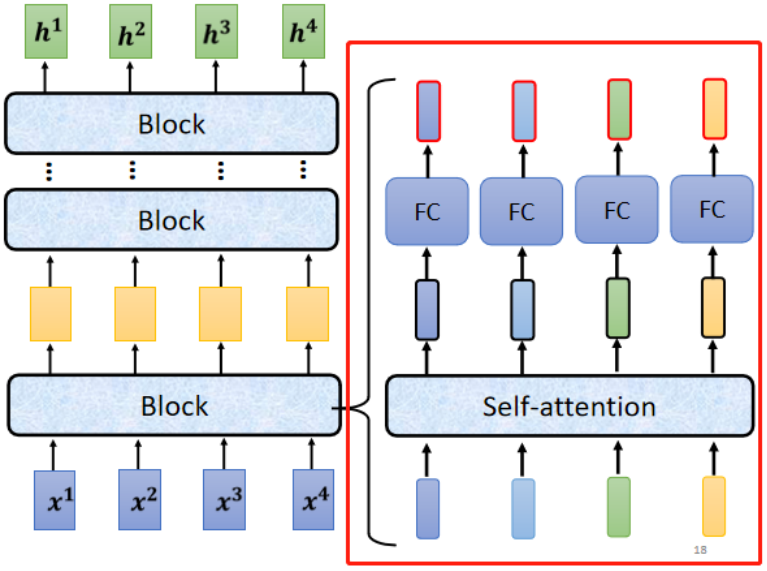
3.1 Encoder

編碼器要做的事情就是給一排向量，輸出另外一排向量。自注意力、循環神經網絡（Recurrent Neural Network，RNN）、卷積神經網路都能輸入一排向量，輸出一排向量。**Transformer 的編碼器使用的是自注意力**，輸入一排向量，輸出另外一個同樣長度的向量



3.1.1 內部剖析

Encoder 中會分成很多的 block，每一個 block 都是輸入一排向量，輸出一排向量。最後一個 block 會輸出最終的向量序列



Encoder 的每個 block 並不是神經網路的一層，在每個 block 中，輸入一排向量後做 Self-attention，考慮整個序列的訊息，輸出另外一排向量。接下來這排向量會進到 FC，輸出另外一排向量，這一排向量就是一個 block 的輸出

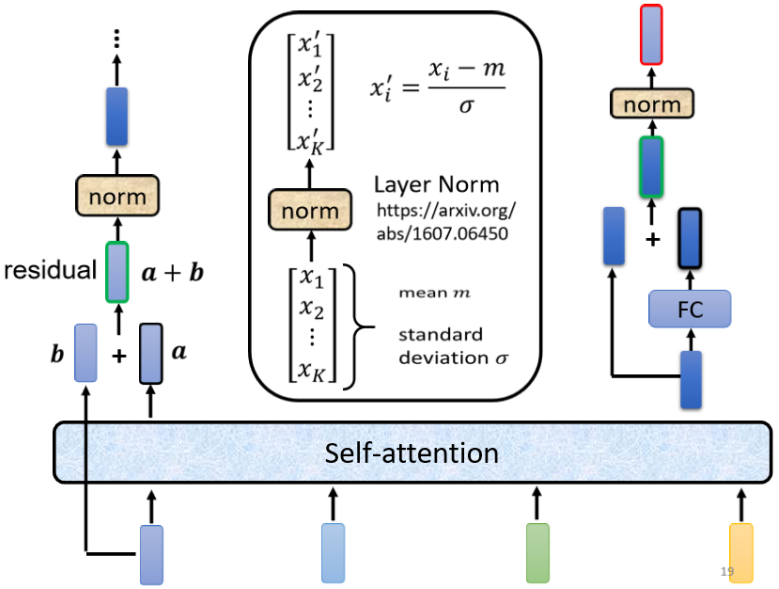
3.1.2 Transformer 的 Encoder

Transformer 做的事情是更複雜的，因 Transformer 加入了 **residual connection** 和 **layer normalization** 的設計

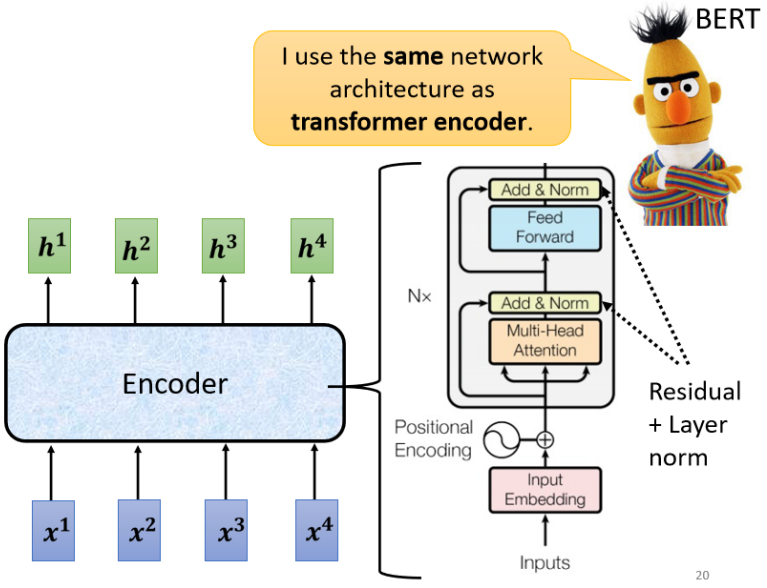
步驟：

1. 考慮全部向量經由 Self-attention 得到輸出向量 a，向量 a 加上其輸入向量 b 得到新的輸出，稱為 residual connection

2. 計算輸入向量 $a+b$ 的 mean 和 standard deviation，做 layer normalization
3. 得到的輸出作為 FC 的輸入，FC 輸出結果和原輸入做 residual connection，再做一次 layer normalization 得到的輸出就是 Transformer Encoder 中一個 block 的一個輸出向量



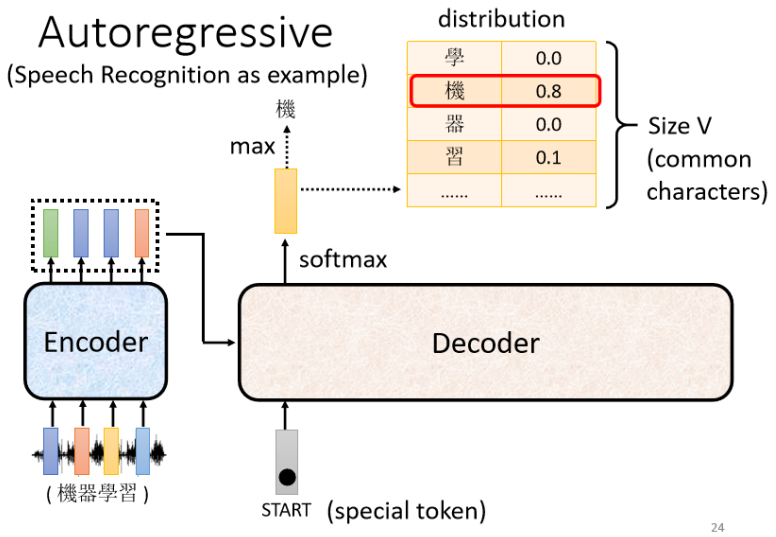
N 表示 N 個 block。首先在輸入需要加上 positional encoding。Multi-head attention 就屬 Self-attention。過後再做 residual connection 和 layer normalization，接下來還要經過 FC，接著再做一次 residual connection 和 layer normalization。如此是一個 block 的輸出，總共會重覆 N 次



3.2 Decoder

3.2.1 autoregressive（AT）

以 encoder 的向量為輸入，並加上特殊的 token 符號 <BOS>（Begin Of Sequence）。在 NLP 中，每一個 token 都可以用一個 one-hot vector 表示，其中一維是 1，剩餘都是 0

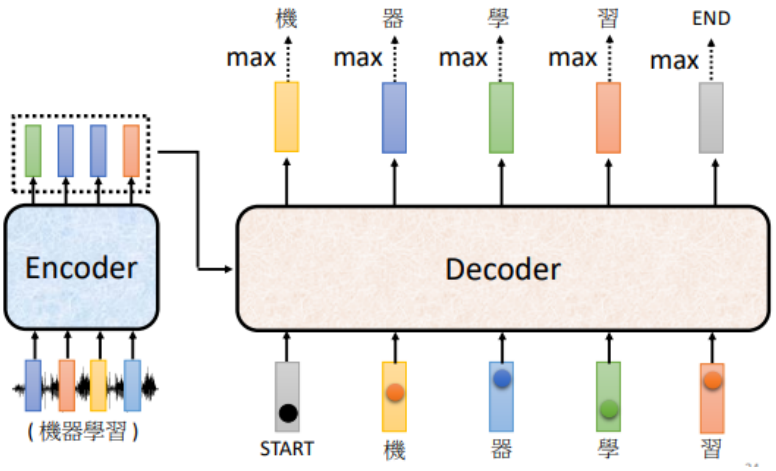


步驟：

1. 向 decoder 輸入 encoder 產生的向量
2. 在 decoder 可能產生的文字裡面加上特殊 token <BOS>
3. decoder 輸出一個向量（長度與 vocabulary size 一樣），隨後通過 softmax，挑選分數最高的一個字作為最終的輸出

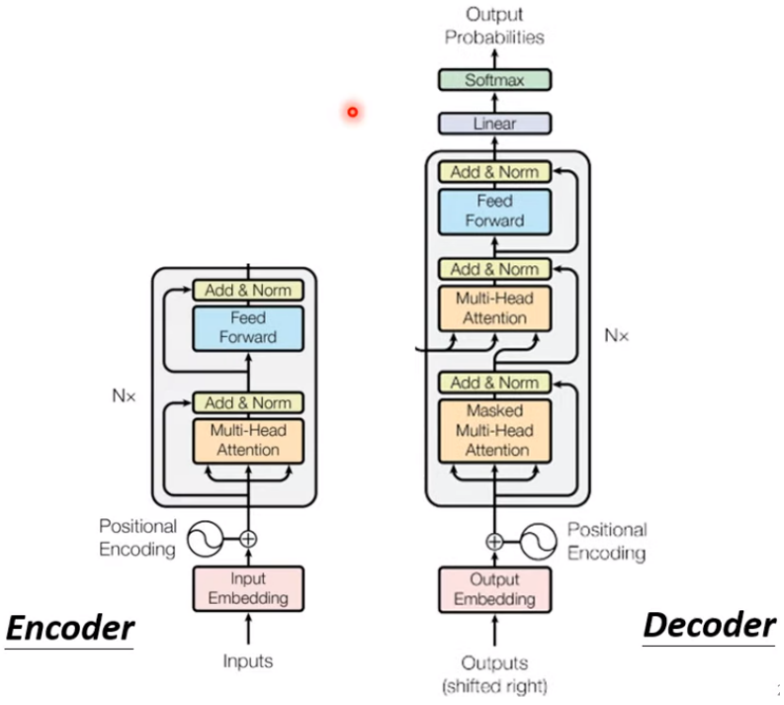
> **vocabulary size：取決於輸出的單位。比如輸出中文，則 size 是中文方塊字的數目**

4. 將 3. 的輸出作為 decoder 新的輸入
5. 重複步驟 3. 和 4.
6. 從 vocabulary 中挑到 <EOS> token，讓 decoder 停止



decoder 的輸入是它在前一個時間點的輸出，其會把自己的輸出當做接下來的輸入，因此當 decoder 產生一個句子時，有可能看到錯誤的東西，造成 **error propgation**

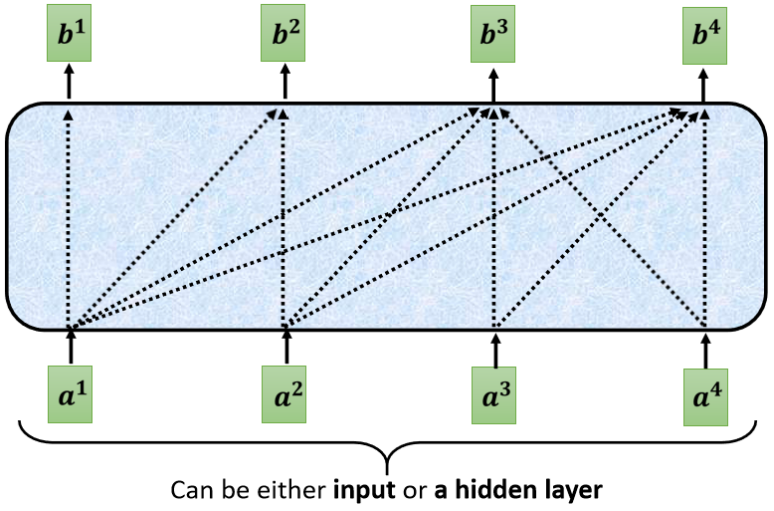
3.2.2 Transformer 的 decoder



除了中間的部分，encoder 跟 decoder，並沒有甚麼差別。最後會再做一個 softmax，使得它的輸出變成一個機率分布，最主要差別是 decoder 的第一個 self-attention 是使用 **masked multi-head attention**

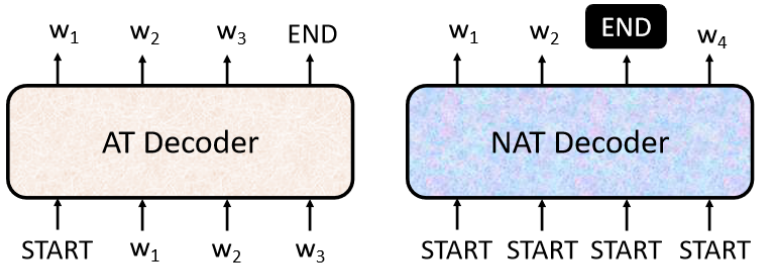
M**asked Multi-Head Attention：**

產生的**輸出並不考慮"右邊"的部分**，原因是因為 decoder 輸出原理是順次產生



3.2.3 non-autoregressive（NAT）

NAT 不是依次產生，而是一次吃一整排的 <BOS> Token，把整個句子一次性產生出來



- How to decide the output length for NAT decoder?
  - Another predictor for output length
  - Output a very long sequence, ignore tokens after END

問題：如何確定 <BOS> 的個數？

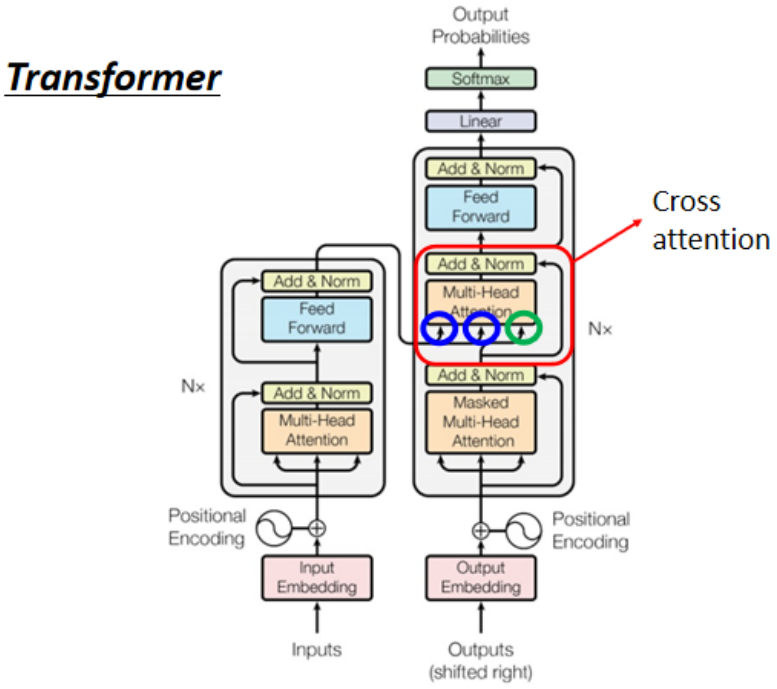
- **另外訓練一個 classifier，**吃 Encoder 的輸入，輸出一個數字，代表 decoder 應該要輸出的長度
- **給很多個 <BOS> 的 token**，例如 300 個 <BOS> 然後就會輸出 300 個字。**什麼地方輸出 <EOS> 表示這個句子結束的點**

NAT 的好處：

- 平行化：
  NAT 的 decoder 不管句子的長度如何，都是一個步驟就產生出完整的句子，所以在速度上 NAT 的 decoder 比 AT 的 decoder 要快
- 容易控制輸出長度：
  例如語音合成有一個 classifier 決定 NAT 的 decoder 應該輸出的長度，並以此調整語音的速度。如果要讓系統講快一點，那就把 classifier 的 output 除以二，如此講話速度就變兩倍快

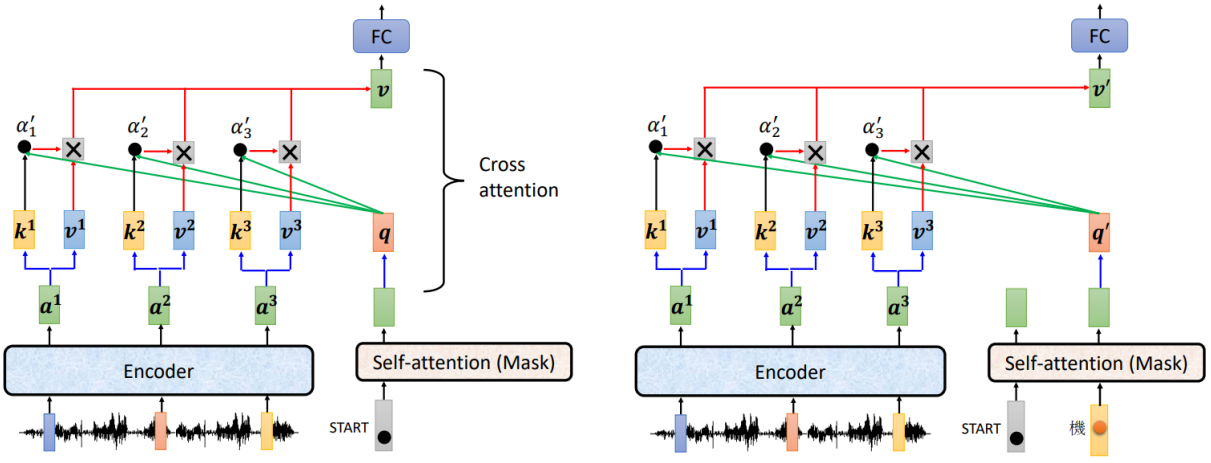
NAT 的 decoder 的 performance 往往都比 AT 還差，原因：**Multi-Modality**

3.3 Encoder-Decoder 的 CrossAttention



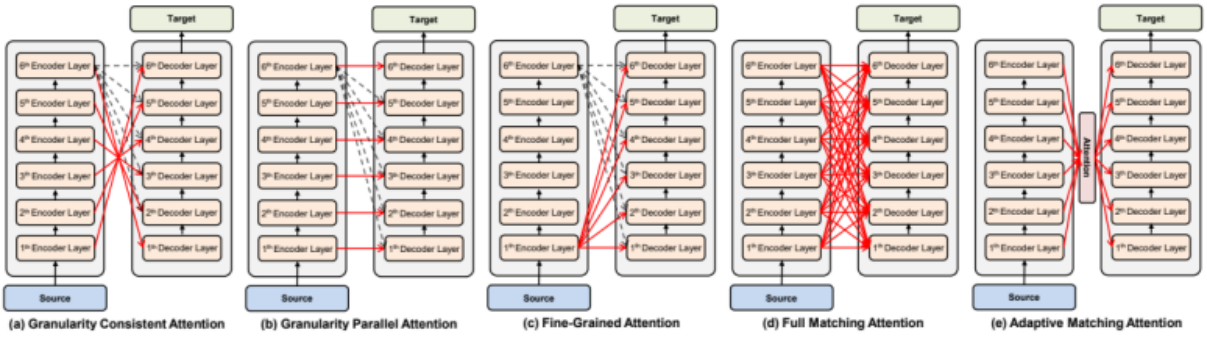
兩個輸入來自 Encoder（Encoder 提供兩個箭頭）， Decoder 提供了一個箭頭

細節：



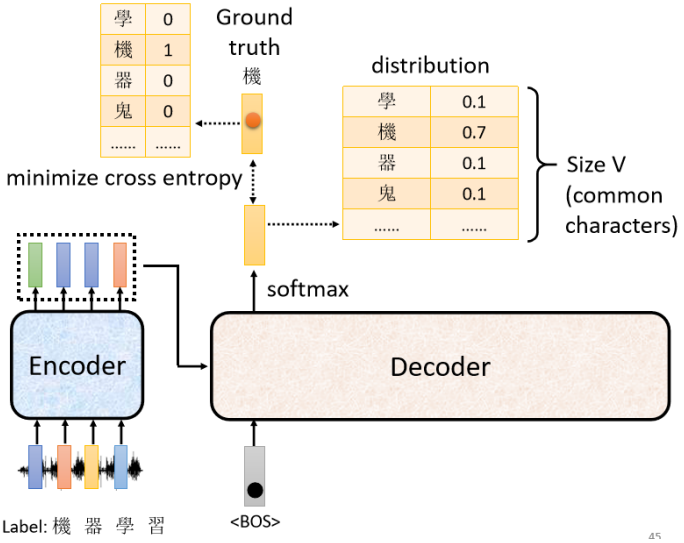
1. encoder 輸入一排向量，輸出一排向量 a^1, a^2, a^3，產生 k^1, k^2, k^3 及 v^1, v^2, v^3，
2. decoder 輸入 <BOS> 經過 self-attention（masked） 得到一个向量，乘上一個矩陣得到 q
3. 利用 q, k 計算 attention 的分數，並做 normalization，得到 $\alpha'_1, \alpha'_2, \alpha'_3$
4. $\alpha'_1, \alpha'_2, \alpha'_3$ 與 v^1, v^2, v^3 做 weighted sum 得到 v
5. 將 v 輸入至 FC 做接下來的任務

總而言之，decoder 就是產生一個 q，去 encoder 抽取訊息出來當做接下來 decoder 的 FC 的 Input



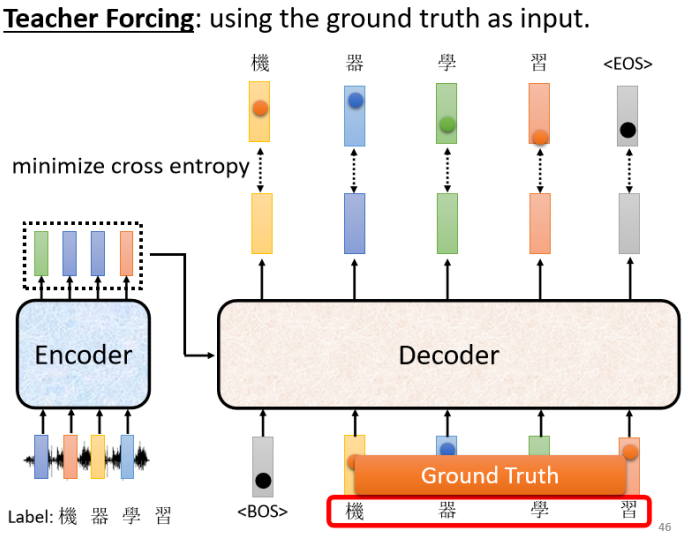
decoder 可以看 encoder 中的許多層而不一定只是最後一層：*Rethinking and Improving Natural Language Generation with Layer-Wise Multi-View Decoding*

4. Transformer 訓練過程



訓練資料：一段音頻與對應的文字，文字為 one-hot encoding 的向量

訓練過程：decoder 輸出的是機率分布，可以通過輸出的機率分布與 ground truth 之間的計算 cross entropy 並求梯度實現優化，使 cross entropy 的值越小越好



注意：

在訓練 decoder 時，輸入的是**正確答案**（ground truth）而不是自己產生的答案，稱作 **Teacher Forcing**

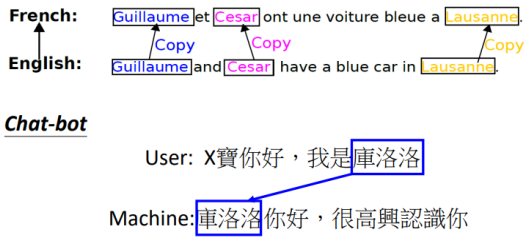
5. Seq2Seq 模型訓練技巧

5.1 Copy Mechanism

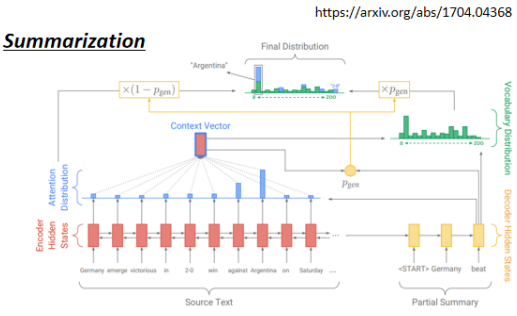
decoder 沒有必要自己創造輸出，它需要做的事情是從輸入的資料中複製一些東西出來，而不是“創造詞彙”

舉例：

1. Chatbot



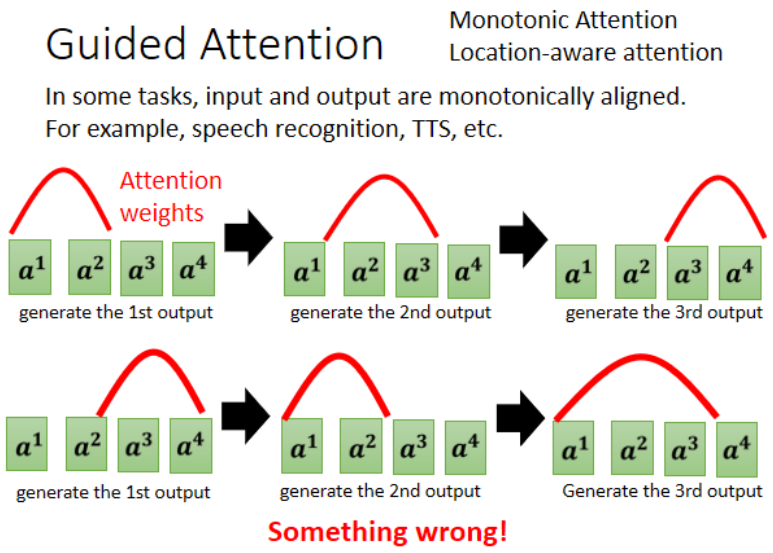
2. Summarization



5.2 Guided Attention

目的：

強迫模型一定要把輸入的每一個東西通通看過（如 TTS），強迫 attention 要有固定的方式



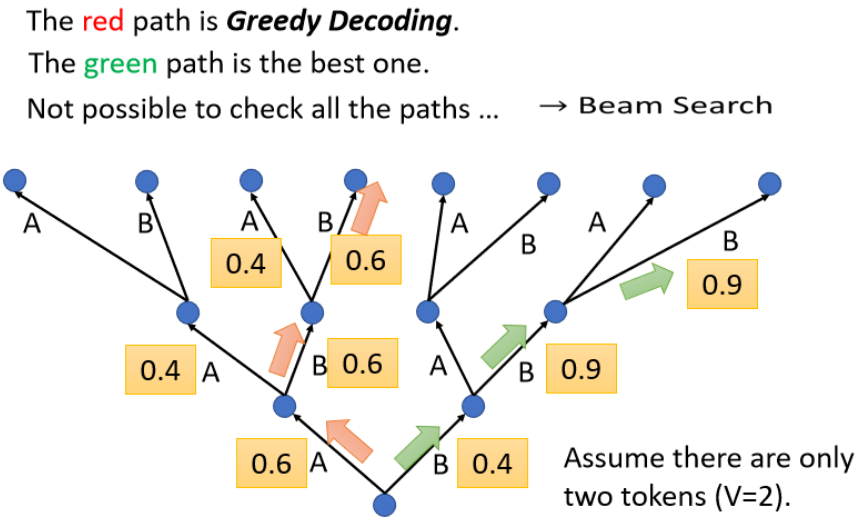
動機：

Seq2Seq Model 有時候 Train 會產生莫名其妙的結果，比如漏字，例如：對語音合成或者是語音辨識來說，我們想像中的 attention，應該要由左向右如上方的圖，但有可能模型跳著看，就如上方的圖

更多資訊：Monotonic Attention、Location-aware Attention

5.3 Beam Search

每次找分數最高的詞元來當做輸出的方法稱為 greedy decoding。紅色路徑就是通過 greedy decoding 得到的路徑。 但貪心搜索不一定是最好的方法，紅色路徑第一步好，綠色路徑一開始比較差，但最終結果是綠色路徑比較好



beam search 用比較有效的方法找一個估測的 solution、一個不是完全精準的 solution，這個方法有時候有用，有時候沒有用，因為找出分數最高的路不見得比較好，取決於任務本身的特性

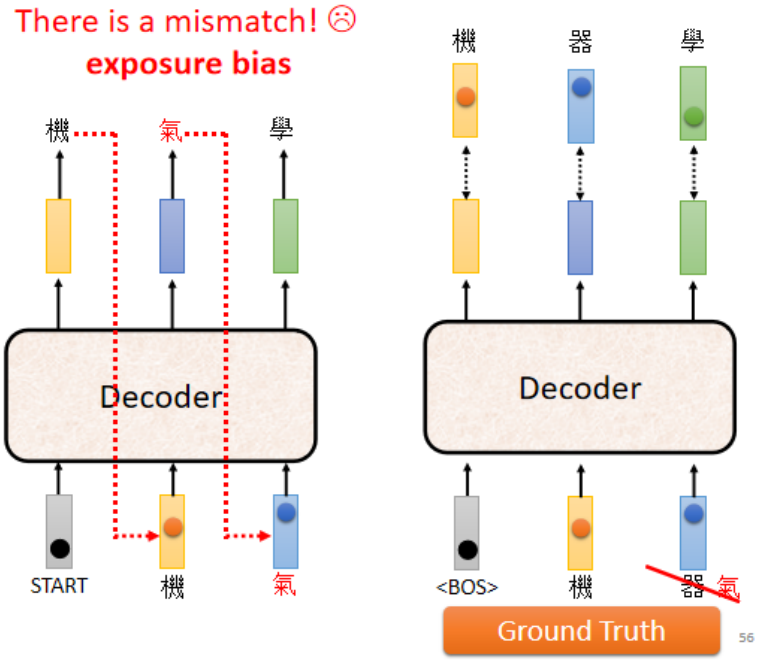
假設任務的答案非常明確，比如語音識別，說一句話，識別的結果就只有一個可能。對這種任務而言，通常 beam search 就會比較有幫助；但如果**任務需要模型發揮一點創造力，beam search 可能比較沒有幫助**

5.4 加入 Noise

語音合成模型訓練好以後，測試時要**加入一些 noise**。用正常的解碼的方法產生出來的聲音聽不太出來是人聲，產生出比較好的聲音是需要一些隨機性的，所以加入一些隨機性的結果反而會比較好

5.5 Scheduled Sampling

測試時，decoder 看到的是自己的輸出，因此它會看到一些錯誤的東西。但是在訓練的時候，decoder 看到的是完全正確的，這種不一致的現象叫做 **exposure bias**



問題：

因為 decoder 從來沒有看過錯的東西，它看到錯的東西會非常的驚奇，接下來它產生的結果可能都會錯掉，導致一步錯步步錯

解決：

給 decoder 的輸入加一些錯誤的東西，模型反而會學得更好 ⇒ **Scheduled Sampling**