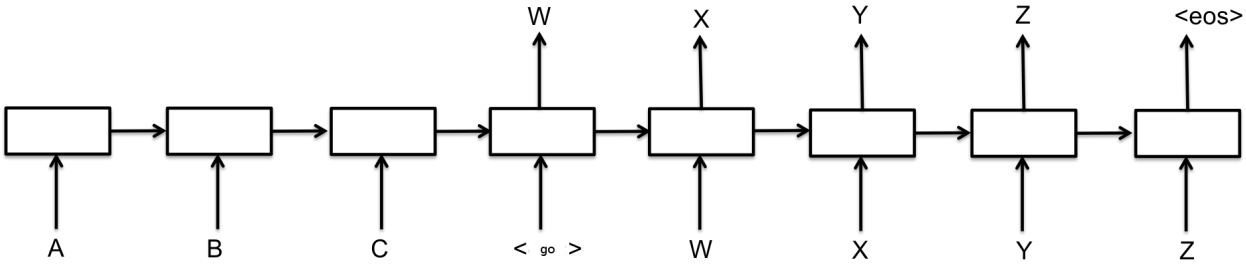
ADLxMLDS HW2 Report

R06922022 資工所碩一 曹燁文

（一）模型描述

1. seq2seq

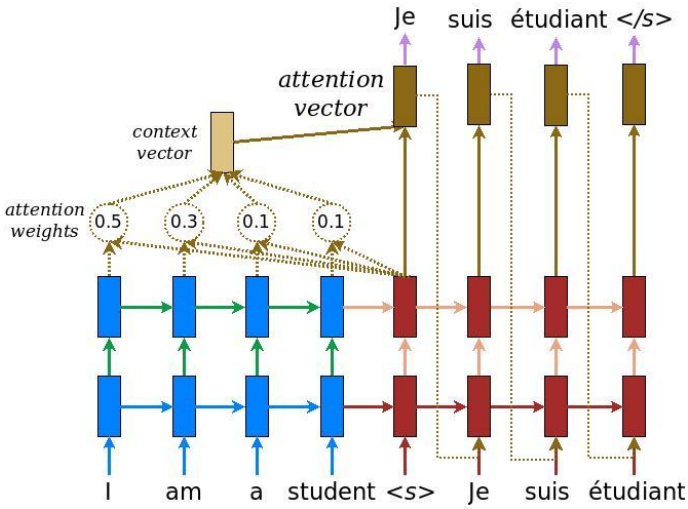
使用 tf.contrib.seq2seq 建立 encoder-decoder，大致概念如下圖：



credit: https://github.com/guillaume-chevalier/seq2seq-signal-prediction

Seq2seq 分為兩階段，training 和 inference，對於 encoder 來說兩階段的做法一樣，ABC 是 encoder 的輸入，在此作業中是固定 timestep 長度為 80 的影片 frame feature 輸入。Decoder 下方的句子稱為 source sentence，上方稱為 target sentence，training 方法是是讀入 encoder 的 final state 以及 source 的第一個字(<go>, or <bos>)，輸出一個 output，將更新之後的 state 傳入下一個 timestep，讀入第二個 source 的字，反覆這些步驟直到 source sentence 結束，最後用全部的 output 和 target sentence 計算 cross entropy loss。Inference 與 training 的不同就在於沒有 target sentence，所以每次 decoder 輸入的字是上一個 timestep output 的字，輸出直到出現<eos>或是規定的長度。程式碼主要參考 https://github.com/tensorflow/nmt 中建立 encoder-decoder 的寫法，所以變數名稱幾乎雷同。

（二）attention (圖片皆來自 nmt 的教學)



使用 tf.contrib.seq2seq 中的 Luong Attention，分別計算 attention weights, context vector 和 attention vector，式子：

$$\alpha_{ts} = \frac{\exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'=1}^{S} \exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)} \quad \text{[Attention weights]} \quad (1)$$

$$\boldsymbol{c}_t = \sum_s \alpha_{ts} \bar{\boldsymbol{h}}_s \quad \text{[Context vector]} \quad (2)$$

$$\boldsymbol{a}_t = f(\boldsymbol{c}_t, \boldsymbol{h}_t) = \tanh(\boldsymbol{W_c}[\boldsymbol{c}_t; \boldsymbol{h}_t]) \quad \text{[Attention vector]} \quad (3)$$

$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \boldsymbol{W} \bar{\boldsymbol{h}}_s & \text{[Luong's multiplicative style]} \\ \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W_1} \boldsymbol{h}_t + \boldsymbol{W_2} \bar{\boldsymbol{h}}_s\right) & \text{[Bahdanau's additive style]} \end{cases} \quad (4)$$

最後得到 output。其中有另一個 Bahdanau Attention 也有許多人使用，礙於時間關係無法做太多的實驗所以沒有比較兩者的差異。

（三）Improve performance

我嘗試將 encoder 變成 bidirectional、multi-layer，decoder 變成 multi-layer，有得到更好的結果。此外也嘗試 schedule sampling，但是結果沒有變好。使用 tf.contrib.seq2seq 的 beam search，研究了很久雖然了解 api 如何使用但是仍然不知道為何會產生奇怪的結果。

（四）實驗結果與設定

Model	Layer	lr (1e-4)	epoch	attention	schedule sampling	BLEU	New BLEU
LSTM	256x1	1	145	no	no	0.2911	X
LSTM	256x2	1	114	no	no	0.2990	X
LSTM	256x2	1	105	yes	no	0.3022	X
LSTM	256x2	1	23	yes	0.999*	0.2948	X
LSTM	256x2	1	29	yes	0.995*	0.2912	X
LSTM	256x2	1	33	yes	0.99*	0.2907	X
LSTM	512x3	0.8	51	yes	no	0.3087**	0.6715**
LSTM	256x2	1	49	yes	no	0.3129**	0.6832**

* schedule sampling 使用 exponential decay，即每次 step 的 probability 都會乘上該數值。

** 使用新的 bleu_eval.py 計算的結果，會比原先的 BLEU 高一些，最後使用

512x3 而非 256x2 是因為 512x3 產生的句子看起來比較好。
從實驗可以發現加上 schedule sampling 沒有變好，此外我也實驗過 train 到快收斂時加入 schedule sampling 也沒有變好。我還發現句子長度在加了 schedule sampling 之後會變長，不過通常都是增加前面出現過的字，形成一串 loop，例如從 a man is playing a guitar 變成 a man is playing a guitar a guitar。
將模型加大、加深也不一定會讓 BLEU 結果變好，而且 BLEU 似乎不太能真正反應出句子的好壞。我有另外做過實驗不過沒有詳細記錄，將 vocab_size 調大，發現 BLEU 降低，但是產生出來的句子稍微比 vocab_size 調大前通順，不過因為沒有太明顯的進步我就沒有使用調大的 vocab_size。