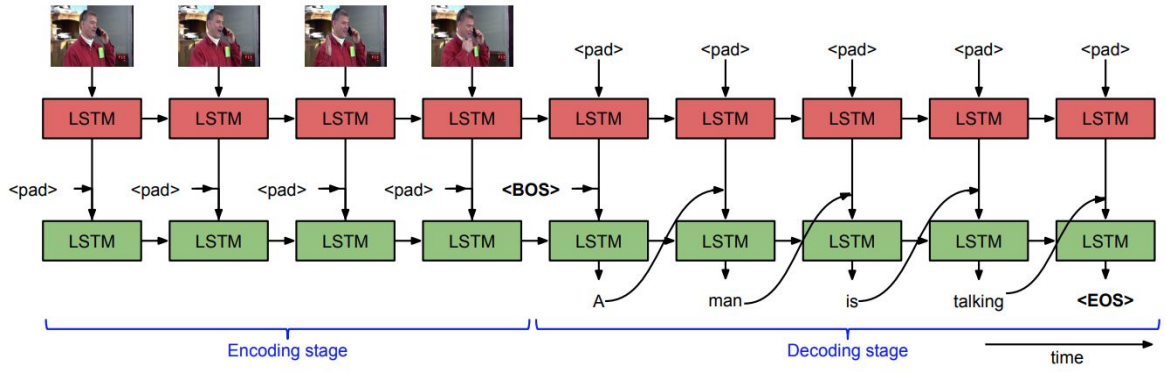
ADLxMLDS HW1 Report

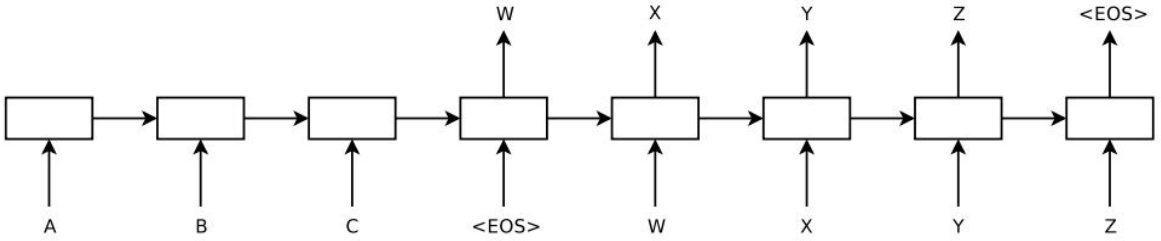
B03902086 資工四 李鈺昇

Model Description

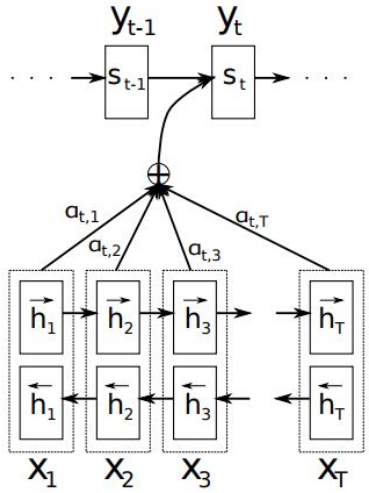
我使用的 model 和投影片中介紹的 S2VT [1] 很像：



不過少了 encoder 的右半段和 decoder 的左半段，因此比較像是 [2] 裡面的樣子：



但是我有加上 attention，所以會增加如 [3] 所描述的機制：



詳細的實作請參考下一段。

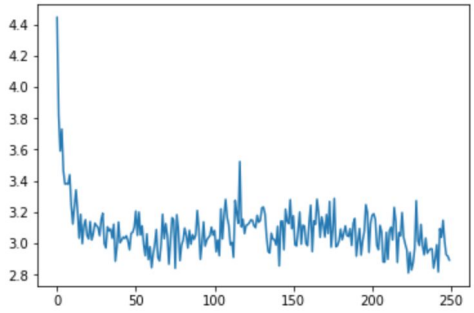
Attention Mechanism

為了實作 attention mechanism，我使用比較靈活的 PyTorch 來實作。

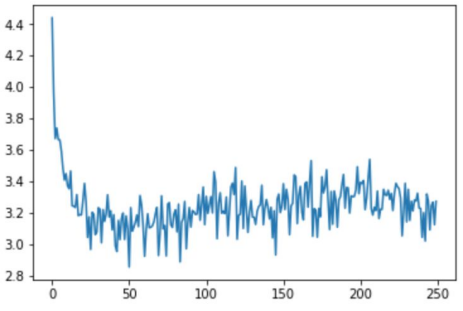
在 decoder RNN 的每一步，RNN cell 會吃進上一步的輸出、上一步的 context，並把輸出和這一步算出的 context 接在一起後經過一個 linear transformation 再找出最大的值的 index，作為這一步的輸出。
而重點就在於 context 的計算：這一步的 context 是經由 encoder 的輸出作為一個可以參考的基準向量 u，並與這一步 RNN 的輸出 v 做一些線性運算後得到一個機率分佈，就是 context。實務上有許多種常用的線性運算，我採用的是 u^TAv，其中 A 是要學的矩陣參數。

分別使用有與沒有 attention 的模型後，可以觀察到如下的差異。

沒有 attention



有 attention



我認為兩者差不多，但這跟預期的結果不太一樣：我的期待是有 attention 的話應該會學得比較快、比較好，不過看起來兩者在這個 dataset 以及我使用的參數下差異不大。

Improving the Performance

最原始的設定是

(1) RNN 採用 256-hidden-size GRU，SGD 跑 100000 epochs，
(2) 使用固定的 scheduled sampling rate = 0.5（隨機決定），並且
(3) 直接把 decoder 輸出的結果原封不動作為答案。

這樣的結果用第一種 BLEU 算法就有 0.2524，直接過了 baseline！

後來我有分別嘗試

(1) RNN 把 256 改成 512、但是跑的 epochs 數量減半、
(2) 使用隨著 epoch 線性變化的 scheduled sampling rate（一開始都用 ground truth，線性變化到最後都用 decoder 上一步的輸出）、
(3) 對 decoder 輸出的結果做一些處理之後才作為答案，

，以下表格有列出一些比較顯著的情況。

	BLEU 算法一	BLEU 算法二	算法一 過 baseline
原始設定	0.2524	0.5198	有
(3)	**0.2640**	**0.6158**	有
(2) + (3)	0.2558	0.5936	有
(2)	0.2429	0.4760	無
(1) + (2) + (3)	0.2618	0.6130	有

其中 (1) 的表現沒有變好，但是我想是因為 epochs 少了一半的關係；如果兩者 epochs 一樣多的話，在沒有嚴重 overfitting 的情況下我認為應該是會變好的。

而 (2) 的表現也沒有變好，這點我認為還滿值得研究，因為理想上應該是線性變化可以讓它學得比隨機還要更好，但我這次並沒有找到原因。

最後，(3) 的「處理」是指在不影響大致結果的情況下，把結尾的一些東西刪掉。我有分別試過把所有結尾的 “a” 和 “.”（句點）都拿掉；以及拿掉句點後再把結尾的 “a” 刪到最多只剩一個。
會想這麼做是因為經過觀察發現這兩者最長出現在句尾並重複很多次，而第二種做法的主因是 “a” 的情況比較特別，例如 “a woman is eating a a . . .” 這種句型，雖然最後的 “a” 乍看之下可以全部刪除，但其實動詞 “eating” 後面有極高的機率會接上 定冠詞+名詞 的組合，因此保留一個 “a” 反而是合理的。
由於第二種做法的分數都比第一種高，表格的 (3) 都是指第二種做法。

Experiments and Settings

首先，我一開始為了減少變因，只採用

** 正在跑 bidir
** 正在跑 pure teacher forcing

Packages and Versions

使用 pip3 freeze > requirements.txt 之後，找出比較重要的幾個如下：

arrow==0.10.0
numpy==1.13.3
torch==0.2.0.post3

References

[1] Sequence to Sequence – Video to Text
[2] Sequence to Sequence Learning with Neural Networks
[3] Neural Machine Translation by Jointly Learning to Align and Translate