HW2 Report

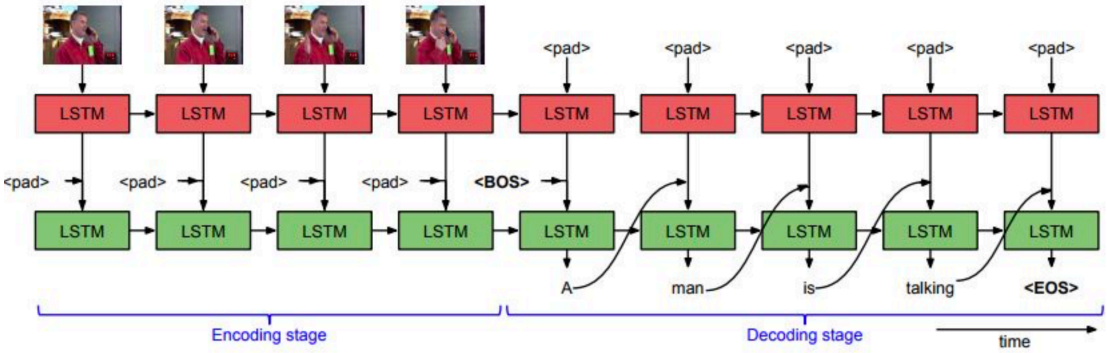
學號:R06942074
系級: 電信所碩一
姓名:李宇哲

Model description:

Platform: Tensorflow

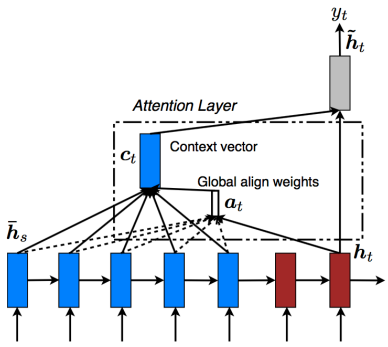
基本上我的架構主要是參考作業投影片上的架構為主。如下圖，在 encoder 的部分我使用兩層 Basic LSTM cell，因為影片的長度有 80 個 frame，所以 encoder 會傳遞 80 次 state 參數，圖片的資料會先經過一層 projection(維度 4096→ 1024)讓影片的檔案降維才會進去第一個 LSTM cell。

在 Decoder 的部分我也是使用兩層的 LSTM，sequence size 設為 20，多於 20 就會被截掉。然後每一個 cell predict 的 output 都會成會下一個 LSTM cell 的第二個 input，如下圖所示。然後 decoder 的第一個 LSTM cell 的 input 如果沒有做 attention mechanism 的話原則上是給 padding，以下會再細講我如何實作 attention model。



Attention mechanism:

參考論文：*Effective Approaches to Attention-based Neural Machine Translation* 可以發現 Attention based model 可以分為 global attention model 和 local attention model，不過 global attention model 是比較廣為使用的 mechanism。如下圖所示，attention model 會以前一個 decoder hidden state 去和 encoder 的 output 做 correlation 也就是公式 8 的 score 的算法，然後從公式 7 的 softmax 算出權重分佈，在利用這個權重去取 encoder output hidden state 作為 decoder 下一個 input(Ct)。



$$a_t(s) = \text{align}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) \quad (7)$$
$$= \frac{\exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'} \exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)}$$

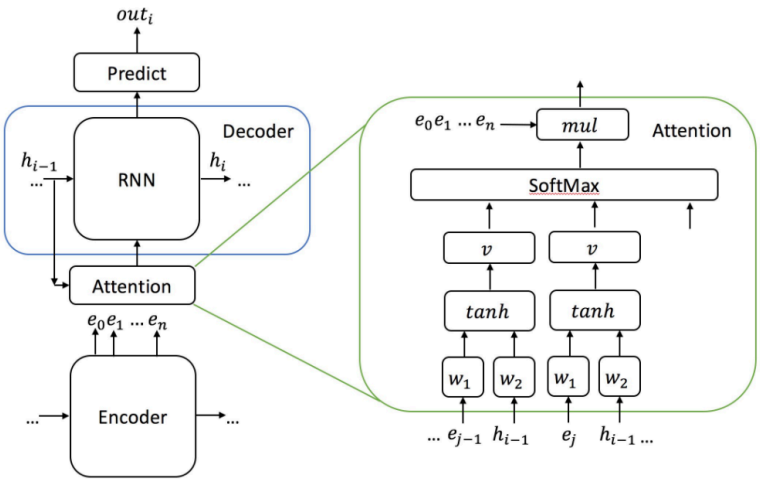
Here, score is referred as a *content-based* function for which we consider three different alternatives:

$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \bar{\boldsymbol{h}}_s & \textit{dot} \\ \boldsymbol{h}_t^\top \boldsymbol{W}_a \bar{\boldsymbol{h}}_s & \textit{general} \\ \boldsymbol{W}_a [\boldsymbol{h}_t; \bar{\boldsymbol{h}}_s] & \textit{concat} \end{cases} \quad (8)$$

My implementating details:

我按照下圖去用 tensorflow 實作 attention mechanism，比較重要的三行 code 如下，a 為前一個 hidden state 去乘上一個 W2 的 matrix 然後 copy 成與 encoder 數目一樣（80 個），然後 b 為 encoder 的每一個 state 也去做和 W1 做 matrix 的 projection，最後在 C 的地方完成 tanh 的 activation 和 softmax 的 weighting。

```
###################### Implementation of Attention Based Model ################################
    a = tf.tile(tf.expand_dims(tf.matmul(state1, self.attenW1),1),[1,self.n_video_lstm_step,1])
    b = tf.reshape(tf.matmul(tf.reshape(attention_concate,[-1, self.dim_hidden]), self.attenW2),[self.batch_size, self.n_video_lstm_step
    c = tf.nn.softmax(tf.tanh(tf.add(a,b))*self.V,1)*tf.reshape(attention_concate,[self.batch_size, self.n_video_lstm_step,self.dim_hidd
```

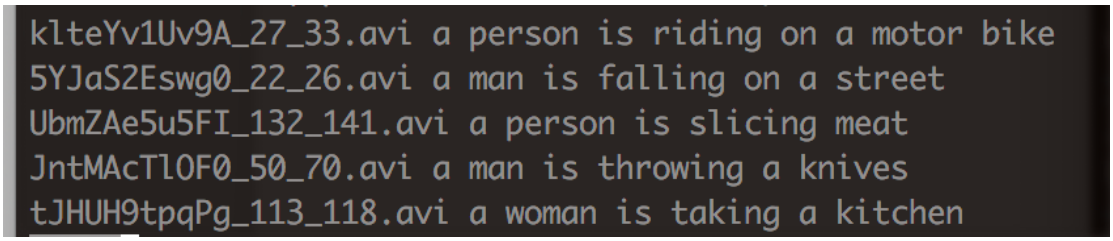


拿到當下的 attention state input 後與第一個架構的 padding 取代，所以 decoder 的 input 都會是 attention model 的 result。

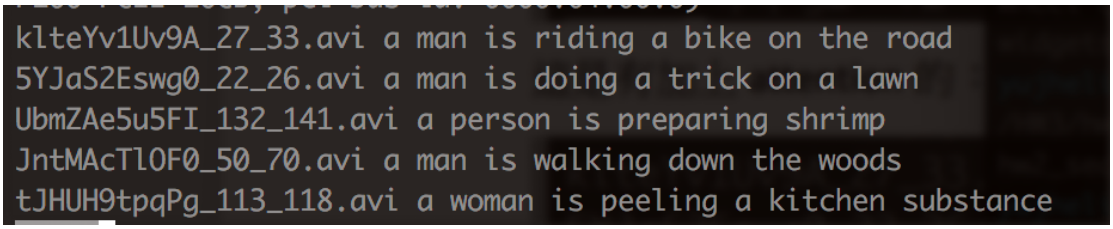
Compare and analyze the results of models with and without attention mechanism:

說到 attention model，我覺得 attention model 似乎有提升判斷的合理性，但是在 bleu score 上並沒有顯著的提升，甚至還有下降的狀況，如圖：

這是沒有 attention 的： blue score 0.28



這是有加上 attention 的：blue score 0.27



True labels are:

1. "A man is riding a motorcycle."
2. "The girl is jumping rope."
3. "A woman is preparing fish."
4. "Boys are running in the wood."
5. "A woman is juicing a lemon."

其實可以發現加上 attention 的 model 的 score 竟然是呈現下降狀態， 可是預測出來的句子的確有比較接近答案，尤其是第二個跟第四個答案，雖然要完整預測 ground true 很困難，但是 attention based 的 model 有稍微更合理的趨勢。但是我自己也懷疑 attention model 到底有沒有用，畢竟加上 attention 後 loss 並沒有下降的趨勢。

How to improve your performance：

我增加 LSTM 的 dimension 到 1024，然後增加 training 的時間，大概 2000 epoch 用 p100 大概要跑半天。每 200 epoch 去存一次 model，然後挑最好的。不過最好的 model 通常都不是 bleu score 最好的 model，所以我都是去把 predict 印出來用肉眼比較。

Experimental results and settings:

我的實驗 setting 如下

Tensorflow 1.3.0

Image data dimension: 4096

LSTM cell dimension: 1024

Encoder length: 80 steps

Decoder length: 20 steps

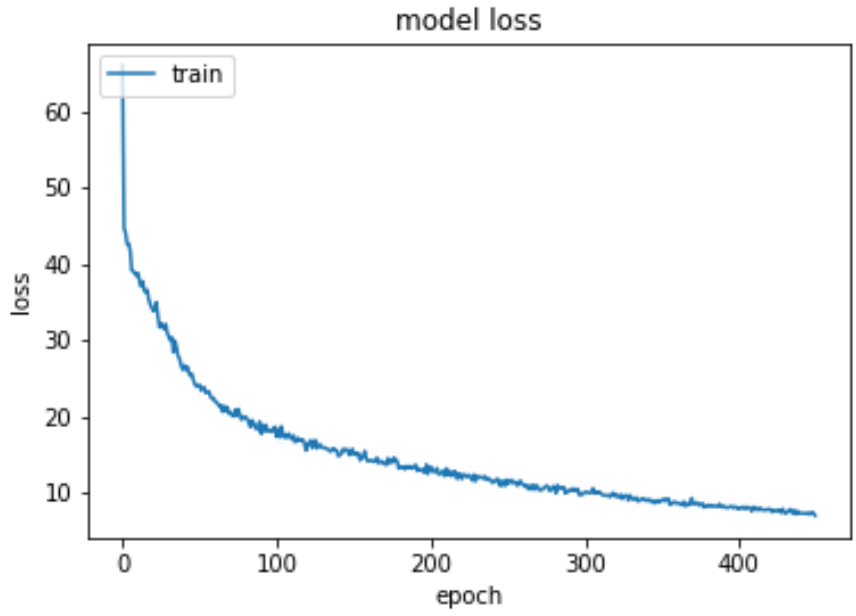
Vocabulary size: 6200 + <PAD>+<EOS>+<BOS> +<UNK>

Learning rate: 0.05

Optimizer: SGD

Using Attention based model and teacher forcing

下圖是前 500 個 epoch 的結果，可以發現 loss 基本上容易維持在 5~10 附近，我的 training 過程會隨機從每一個 caption dict 抽出一個 caption，所以同一部影片會有多個 label 但是不會在同一個 epoch 同時 train 到，可以推測 loss 很難降到 2 以下甚至 1 以下可能是因為我用 random 選擇的關係。但是如果不用 random 而選擇某一個 caption，反而會 overfitting 一下子降到 0.0XXX，可能連 attention model 都沒有 tune 到參數。



不過在我試過 schedual sampling 後，我發現不管是採用 linear 或是 exponential decay 還是維持一個機率，反而不能使 model 減少 loss，還會增加 model 的 loss 甚至還會飆到 20 左右下不來，然後回去看 predict 的結果也沒有比較好，甚至會 predict 出非正常句子。