MLDS: HW2

November 19, 2017

Professor Hung-Yi Lee

電機四 B03901018 楊程皓

Model Description

我的 Model 基本上跟上課所教的 sequence to sequence 沒有什麼差異，以上下兩個 LSTM (以下簡稱 LSTM1, LSTM2) 為主體。在 encoder (輸入一段影片 80 frame 的畫面) 階段，把 input 在連續 80 time steps 中丟入 LSTM1, 將 LSTM1 output 合併 padding 作為 LSTM2 intput，並忽略 LSTM2 output。在 decoder 階段，LSTM1 input 持續給 padding，並將 LSTM1 output 連同 caption label (在 testing 階段就換成上一個 time step 的 LSTM2 output: caption prediction) 作為 LSTM2 input。

然而雖然 input video 都是 80 frame, 我們不需要做特別的 video mask，產出的 caption 卻很可能有長度差別很大的句子。故在 training 時，我預先統計了一下 caption 長度的平均值為 7.7，標準差為 3.3，最大值為 40，最後我就把 decoder 最長設為 40，並依據不同的 caption label, 將實際長度之後的 time steps 的 caption mask 設為 0, 即不計算那些 time step 的 loss。

在 training data 方面，因為同一個音檔有很多不一樣的 caption labels 都算是正解，故同個音檔就被複製成 caption labels 個數筆不同的 training data 進去 train，彼此沒有關連，在 model 的實作方面也比較方便。

Attention Mechanism

- How do I implement:
  我嘗試了兩種架構。
  第一種：基於我一開始的架構，將 LSTM2 在 encoder 階段的 80 個原來棄之不用的 output 做一個線性組合 (線性組合的參數之後會跟隨整個 model 一起學出來)。在 decoder 階段，將 attention 產出來的 output 取代原來的 padding 作為 LSTM1 的 input，而 LSTM2 一樣接收 LSTM1 output 與前一個 time step 的 caption label 或 caption prediction 的合併作為 LSTM2 input。這個做法是在網路上找其他人的做法，並小作修改而來的，但後來想想這樣做其實不太合理，因為我認為 attention 應該把 decoder 階段每個 time step 的 state 或 prediction 作為 input 再決定 attention 要給什麼值，故我實作了第二種做法。Github branch: attention1.

  第二種：仿照 hw2 slides 提供的 attention 論文上的做法，也是將 encoder 階段的 LSTM2 output 取出來作為 attention base，對於 LSTM1, LSTM2 input, output 都不做任何修改，但 LSTM2 output 並不直接作為 prediction，而是定義一個 $score(h_t, \bar{h_s}) = h_t^\top W_a \bar{h_s}$，並讓 W_a 跟著 model 一起被學出來，即論文中的 global attention model。
- Compare and analyze 第一種：在兩種 bleu 計算方式下的平均值約為 (0.25, 0.54) 左右，比未加上 attention mechanism 的結果略差一些，可能原因為這樣的 attention mechanism 其實對

LSTM1 而言並沒有幫助，畢竟只是重複在 decoder 階段餵前面的 outputs 的組合給 LSTM1。

第二種：由於時間緊迫，這個我完全自己想辦法 implement 的方法雖然可以跑，loss 也有照預期下降，跑出來的 prediction 卻都只有一個字或重複的字，很明顯 attention mechanism 有地方寫的不對，prediction 被嚴重 bias 掉了，之後若有機會再使用的話需要再檢查 Tensor 轉換流程。

How to improve your performance

我主要嘗試了三個方向。
第一個是改變 epoch 次數，這個方法比較直觀也比較好做，就是把每個 epoch 都存一次 model, 然後寫個 script 跑結果把 bleu 印出來，實驗結果是其實在大約 5 10 epoch 時 bleu 的值最高，大概到 100 epoch 就會嚴重 overfitting, 產出來的句子也不能看了。
第二個是考慮要不要 shuffle training data，一開始是有 shuffle，後來在考慮是不是同音檔綁在同一個 batch 裡面可能會避免來回震盪的情形，也考慮不要 shuffle，但實驗發現不 shuffle 的結果會變差，故最終的 model 是有 shuffle 的。
第三個是在 training 過程中，decoder 階段餵給 LSTM2 作為 input 的要是上一個 time step 的 caption label 還是 prediction，實驗結果是效果差不多，故最後選擇使用 caption label。

Experimental results and settings

下圖為我所上傳的最佳 model 在不同 num of epoch (1-19) 底下的兩種 Bleu 的值。可以看到在 epoch 5-10 之間有最好的幾個數值，故我的最終 model 選的是 epoch 9 的版本。
其他參數：LSTM hidden units = 256, batch size = 32, learning rate = 0.001, max epoch = 40.