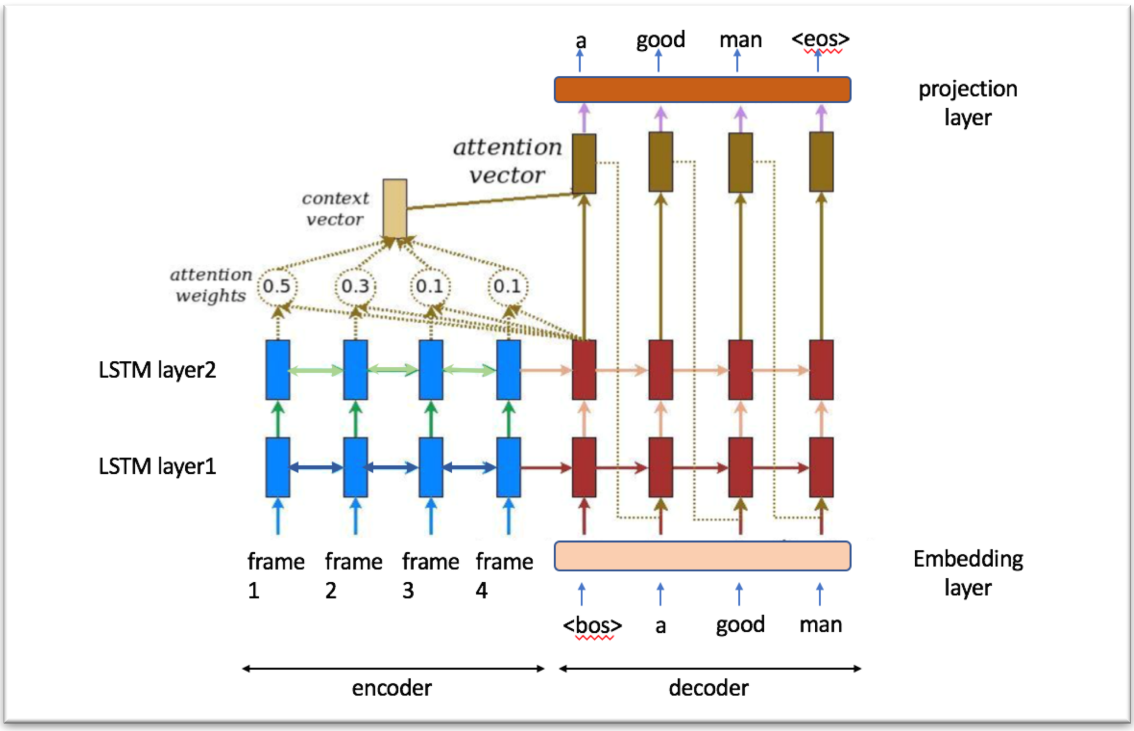
ADL-MLDS hw2

R06922057 梁智泓

Model description



我的 model 結構如上圖所示，分為 encoder 與 decoder，encoder 主要由兩層的 bi-directional LSTM layer 所構成，decoder 則由兩層的 LSTM 所構成，而在 decoder 的 input，我加上了 word embedding layer，讓每個 one hot 的 word vector 能夠用更好的 word vector 所表示，進而提升 performance，而在 encoder 與 decoder 的 output 我加上了 attention layer，在 decoder 的最後再加上 projection layer，將輸出的結果再轉回原本的 one hot word，最後將生成的句子輸出。

Attention Machanism

在 attention layer 這，我使用了 Luong Attention，將 encoder 的 output 餵進 attention layer，算出 attention weight，再透過新的 attention weight 算出 context vector，最後再將 context vector 與 hidden layer 的 output 結合，生成新的結果。

而在實驗上，可以明顯看出加上 attention layer 之後，performance 有明顯的，相關的實驗數據，在 Experiment 部分會再細說。

How to improve performance

為了提升 performance，我做了許多實驗與改變，下面分點敘述

1. **Word occur num threshold**：因為有一些字在 caption 中出現的次數過少，所以我將出現次數過少的字詞去掉，讓 caption 出來的結果不會出現過於稀少的字詞。

2. **Add Schedule Sampling**：讓 training 時 decoder 的 input 機率性的使用自己的上個字的 output 而非 ground true，比較符合 testing 的狀態，因為在 testing 時，我們使用的是 greed 的生成句子，換句話，下一個字的 input 是目前的 output，但實測的結果都是加上之後 performance 有時會比較好，但大部分的時候都比較差，所以最後的 model 我並沒有使用 Schedule Sampling。

3. **Add Attention layer**：如上面的 attention 所述，我加上了 Luong Attention，加上之後 performance 也有明顯的變好。

4. **Bi-directional LSTM**：我將 encoder 的 LSTM layer 換成了 bi-directional LSTM，希望能夠學習到影片的前後關係，而非一個方向的 video sequence，因為影片的每個 frame 之間與文字一樣，具有前後的因果關係，通常會互相影響，尤其是要生成敘述影片的句子時，更時重要，而 performance 也的確有變好。

5. **加大 LSTM 的 hidden layer num**：希望可以透過加大 LSTM 的 hidden layer num，讓 model 可以學習出更為精細的 feature，然而經過實驗後，發現不斷地加大 LSTM 的 hidden layer 數，performance 的差異並不大。

6. **Multi-Layer LSTM**：希望可以透過加多一層 LSTM，變成兩層的 LSMT 進行 training，來學習更為精細的 feature，而實驗結果也可以看到 performance 有明顯變好。

7. **少一點 epoch**：經過實驗之後發現，在 training 的過程中，performance 大概到了 epoch 30 左右就會收斂，後面就會不斷在某個區間之內上下震動，雖然 loss 有降，但 performance 的結果並沒有跟著變好，所以我選擇了 epoch 數比較少的 model，讓 model 不會 overfitting。

Experiment

One layer

	Bleu_score	epoch	Word_num_thres	Sample rate
Simp	0.27469781	200	2	N/A
+sche	0.291813335	200	2	0.3
+sche	0.273272444	200	2	0.7
+sche+atten	0.278943183	200	2	0.3
+atten	0.284728637	200	2	N/A
+atten	0.282581968	200	0	N/A
+bidir+atten	0.290939245	100	0	N/A
+bidir+atten	0.29656169	50	0	N/A

上面的實驗都是做在一層的 LSTM 上面，從實驗可以清楚地看到，原本的 model 加上 schedule sampling 之後，結果就明顯的變好。但 schedule sampling 再加上其他的作法（attention or bidirection LSTM）之後，performance 都會有些許的降低。除此之外，從上面的實驗也可以看到，當 sample rate 越高（越選擇相信 decoder 所生出來的結果）時，training 出來的結果會越差。而在 word num threshold 上面，也可以看出將冷僻的字去除掉之後，結果也有些許的變好。Epoch 的多寡也明顯影響了 performance，將 epoch 從 200 降為 100 和 50 之後，performance 都有明顯的升高，表示 model 在 epoch 200 的時候早已經 overfitting 了。

Two Layer

	Bleu_score	epoch	lstmLayer	LstmDim
+bidir+atten	0.290939245	100	1	256
+bidir+atten	0.29656169	50	1	256
+bidir+atten	0.293839477	100	2	256
+bidir+atten	0.305217069	50	2	256

+bidir+atten	0.302004228	50	2	500
+bidir+atten	0.299941554	50	2	400
+bidir+atten	0.307436754	30	2	400
+bidir+atten+sche	0.294555685	100	2	256
+bidir+atten+sche	0.292592391	50	2	256

從上面的實驗可以看出，將 LSTM layer 從一層變成兩層，performance 有明顯的變好。而增加了 LSTM 的 hidden layer 數之後，performance 未必會變好，因為 model 可能太過複雜，導致無法學出好的 feature。最後也可以看出加上了 Schedule Sampling 之後，performance 都有些許的下降。

README

這次所有的實驗都是在

python 3.5.2+

tensorflow 1.3

keras 2.1.1

Linux

GPU Tesla T80

的環境下進行實驗的與測試