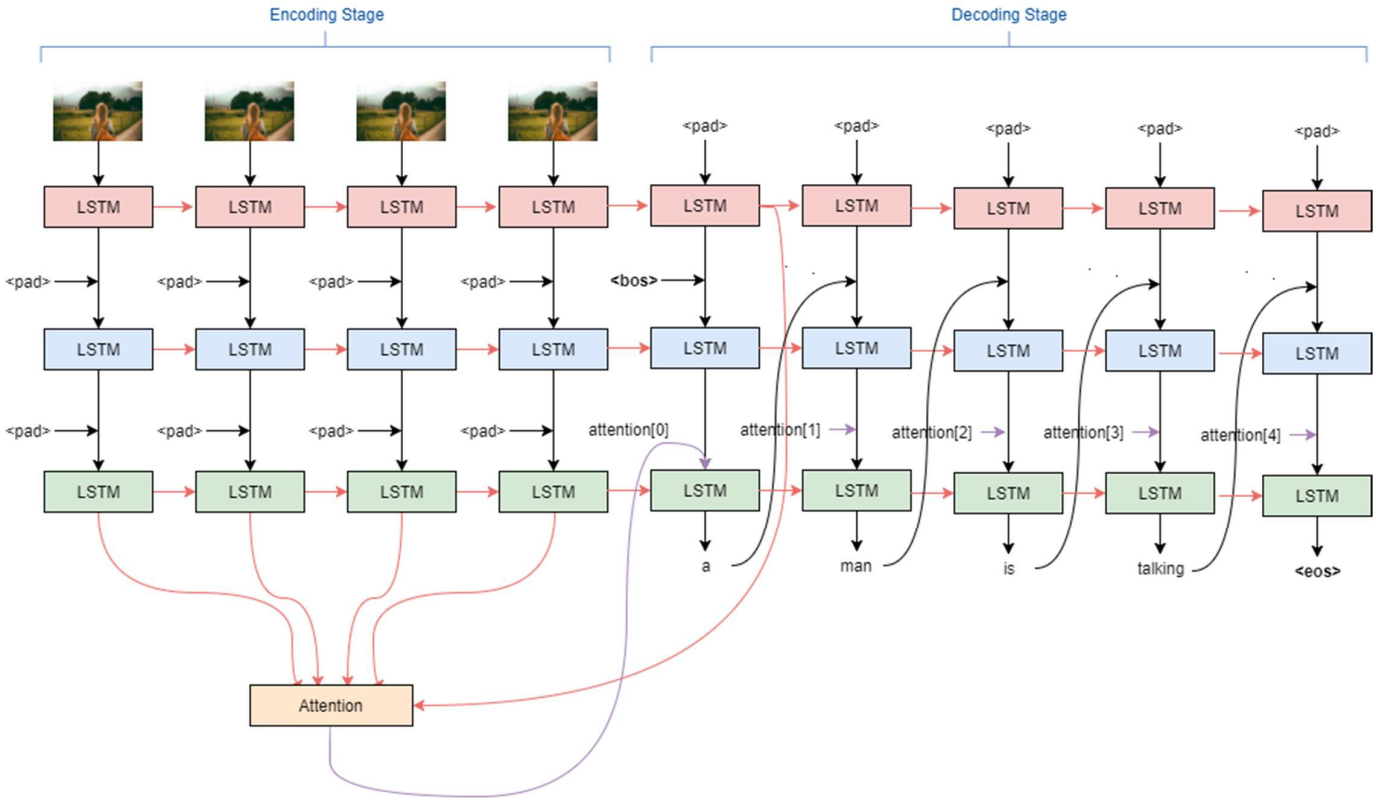
ADL x MLDS Fall 2017

Homework 2 Report

Student: CSIE R06922068 Yu-Jing Lin 林裕景

Model Description

以下是我這次作業所建構的模型架構圖：



使用 Python+TensorFlow 套件進行實作。

參考〈Sequence to Sequence -- Video to Text〉的 S2VT 模型，流程像 seq2seq 分為 encode 和 decode 兩個 stage，先輸入完所有影片的 frames 後再開始預測句子，不同處為 decode 時的文字並非傳到第一層 RNN 而是第二層，讓兩個 RNN 分別學習影片和句子的特徵。除此之外，我也加上 Attention mechanism，使模型在預測時可以 focus 在前面 encode 影片時的不同 time step，以提升預測句子時的精確度。

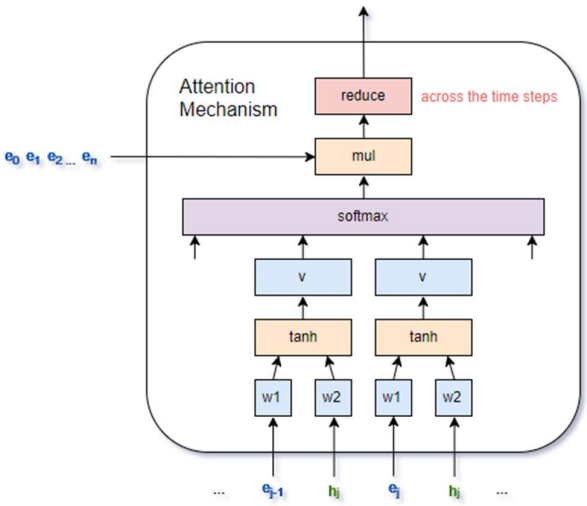
資料 preprocessing 的部分，我做了一個 MSVD dataset loader，便於讀取 training 和 testing 的資料，以及 sentence encoder 來做 word tokenizing 和詞頻統計。

我在 S2VT 模型中的 RNN 使用了 BasicLSTMCell 或 GRUCell，可以用 gradient descent、Adam、RMSProp 三種 optimizer，有 Dropout layer 的設置，可以自訂 outputs 的 keep propagating rate，並 implement 了 scheduled sampling，支持 linear 和 exponential 兩種 decaying rate。

此外，還有依照詞頻調整的 initial projecting bias。並支持 pre-trained word embedding 的 Word2Vec、GloVe 和 FastText 初始化 S2VT 模型中的 word embedding。這兩個方法讓模型在訓練時收斂速度加快。

Attention Mechanism

我使用的 attention mechanism 參考了 Dzmitry Bahdanau 提出的 global attention mechanism，參考了全局的狀態，另外還有人提出 local attention mechanism 但是在這裡我沒有實作。使用在 encoding 時所有第一層的 RNN 之 state h，加上 decoding 時目前第二層的 RNN state h，也就是接受完前一個字詞後的 state h。經過以下架構的 attention mechanism 後輸出作為當前 step 的第三層 RNN 之輸入。



我將 encoding 時第一層 RNN 在每個 time step 的 state h（以下稱為 enc state）乘上一個矩陣 w1，將當前的第二層 RNN 之 state h 乘上一個矩陣 w2 並複製 time step 個，以符合 encoding time step 的數量，然後兩兩做 element-wise 的加法後經過非線性的 tanh 函數，再乘上 v 矩陣，最後對 time step 維度做 softmax，得到的矩陣（或稱做 mask）為每個 time step 該取百分之多少的 enc state。

將這個 mask 與 enc state 做 element-wise 相乘後，即為此階段要輸入給第三層 RNN 的 attention。

其中，w1、w2 和 v 就是這個 attention mechanism 在 training phase 時會訓練到的變數。

How to improve your performance?

起初，我先用一般的 seq2seq 做這次 task，主要使用 TensorFlow 中的 seq2seq 套件，建立基本的 seq2seq 模型，並以這個 model 做 special mission。後來加上 DropoutWrapper、MultiRNNCell、AttentionWrapper 和 BeamSearchDecoder 等等，都用上後，覺得預測的語句似乎沒有很大的提升，於是轉向助教在投影片中提到的 S2VT 模型。

用 TensorFlow 實作了 S2VT 後，發現效果其實也差不多，於是實作了一些其他的機制，如同避免 overfitting 的 Dropout layer，避免 exposure bias 的 scheduled sampling。以及最重要的

attention mechanism，原先是兩層 RNN 架構，attention 吃前一次的 state，但在幾次實驗後，覺得這樣的架構不合適，因而改為三層 RNN 構造。後來也嘗試調整句子的長度和過濾掉出現次數較少的文字。經過以上多次實驗後，我的 model 預測出來的 BLEU 分數明顯上升許多。

而我也發現，由於每支影片對應十幾個描述，若在 training 時只使用同一個句子很容易 overfit 預設的句子，造成預測時大失準，於是我在每個 epoch 時給每段影片隨機抽取一個句子，增加多樣性，並且每個 epoch 還隨機排序訓練影片的順序，避免模型從 data 固定的順序中學到不該學的東西。

再來，我繼續思考可以做甚麼樣的改進，想到既然我們用了 word embedding 在模型裡面，是不是可以匯入外面的 pre-trained word embedding，於是我抽取了 training data 中出現的文字，從 Word2Vec、GloVe 和 FastText 三種方法中拿出對應的 embedding vectors，存成 numpy array，再餵給 S2VT 模型。然而，許多出現的字沒有在 pre-trained embeddings 中，常見的 stop words、極罕見的字，甚至有阿拉伯文字，因此這個 word embedding 得是「可訓練的」。經過幾次實驗後，發現雖然 pre-trained embeddings 可以縮短收店時間，然而到後來的 loss 和 accuracy 都差不多，我猜測是這些 embedding 終究還是會被訓練成差不多的樣子。

Experimental Results and Settings

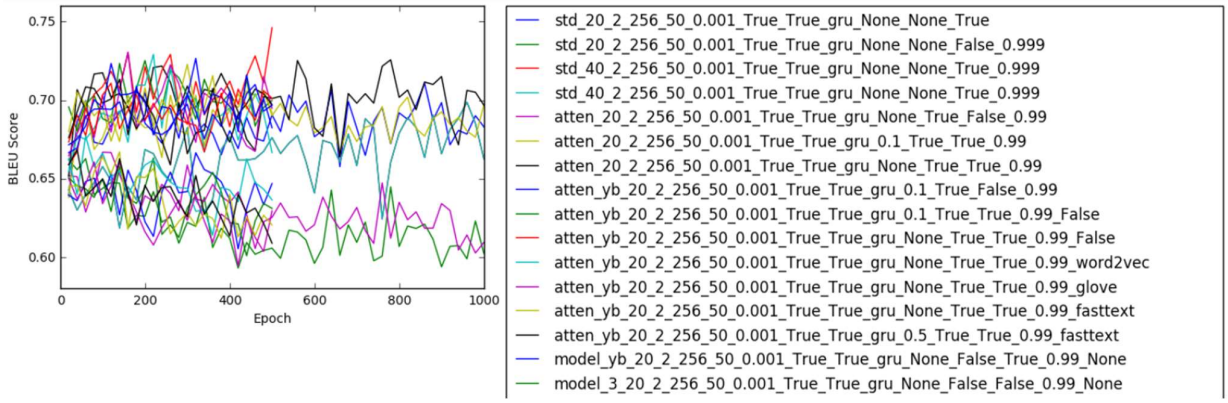
我的實驗環境是：

- CPU: Intel Xeon 處理器 E3-1230 v3
- GPU: GeForce GTX 980
- RAM: 8G
- OS: CentOS Linux 7

使用的工具為 Python 3.5.4 with TensorFlow 1.3.0 and Keras 2.0.7。

我的實驗結果為：

每 20 epoch 一次，以新制 BLEU score 計算，詳細的圖表在 results 資料夾中。



模型名稱的格式為〈模型名稱_label 長度_濾字門檻_RNN 單元數_batch 大小_學習係數_是否隨機 label_是否抽換順序_RNN 種類_dropout 率_是否 attention_是否 sampling_sampling 的遞減係數_sampling 遞減模式_sampling 遞減每 epoch 數_pre-train 的 embeddings〉。每次實驗的 epoch 數有 500 也有 1000，所以圖表中的折線長短不一。

從分數的部分觀察，可以注意到幾個有趣的地方：（可以到 results 資料夾中看分類的結果）

1. Scheduled Sampling 提升

   使用了 scheduled sampling 的結果之 BLEU 分數普遍維持在 0.7 分，有些甚至到 0.74 接近 0.75 分，因為模型透過自己的 output 來判斷下個字，模擬出在預測時沒有 true label 可以參考的情況；而沒使用 scheduled sampling 的結果會因為 overfitting，分數越來越低，到 1000 epoch 左右剩下約 0.6 分。

2. Attention Mechanism 對於分數沒有多大幫助

   每種組合都嘗試過的結果發現 attention mechanism 對分數沒什麼幫助，但是比較過有無使用 attention 的模型所預測出來的結果後，有使用的句子似乎有稍微合理一點，這是分數所看不到的東西。

3. Dropout layer 的沒必要性

   沒有必要使用 Dropout layer，因為 overfitting 在我這次的實驗影響不大，應該說使用 dropout layer 後，結果也沒有比較好。

然而，許多 BLEU 分數高的預測結果，打開一看：（以全部分數最高的為例，單看 special mission）

```
# atten_yb_20_2_256_50_0.001_True_True_gru_None_True_True_0.99_False Epoch 499
klteYv1Uv9A_27_33.avi   => a man is a a on
5YJaS2Eswg0_22_26.avi   => a man is a the
UbmZAe5u5FI_132_141.avi => a woman is the some
JntMAcTlOF0_50_70.avi   => a man is a a a
tJHUH9tpqPg_113_118.avi => a woman is a a a
```

<- BLEU 分數：0.74 多

結論：分數一點都不重要！（？

應該說，分數高確實表示比較多的正解文字被包含在預測內（可能主要是不重要的非關鍵字），但因為描述語句是主觀的東西，單靠比對所謂的標準答案並無法正確的表現出結果的正確性，因此這次作業才會使用 peer review 的方式評分吧！

Reference

Sequence to Sequence Learning with Neural Networks (2014). Ilya Sutskever, Oriol Vinyals, Quoc V. Le. Available at: https://arxiv.org/abs/1409.3215.

Sequence to Sequence -- Video to Text (2015). Subhashini Venugopalan, Marcus Rohrbach, Jeff Donahue, Raymond Mooney, Trevor Darrell, Kate Saenko. Available at: https://arxiv.org/abs/1505.00487.

Neural Machine Translation by Jointly Learning to Align and Translate (2014). Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio. Available at: https://arxiv.org/abs/1409.0473.

Sequence To Sequence Attention Models In DyNet (Github, 2016). Tal Baumel. Available at: https://talbaumel.github.io/attention/.

Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks. Samy Bengio, Oriol Vinyals, Navdeep Jaitly, Noam Shazeer. Available at: https://arxiv.org/abs/1506.03099.

Recurrent Neural Network Regularization (2014). Wojciech Zaremba, Ilya Sutskever, Oriol Vinyals. Available at: https://arxiv.org/abs/1409.2329.