MLDS HW2 Report

組別：我才是真的Baseline
組員：f03942038鍾佳豪 / r05942102王冠驊 / d05921018林家慶 / d05921027張鈞閔

Environment

OS: Windows 10
CPU: Intel Xeon CPU E5-1630 v3 @ 3.7 GHz with 64GB RAM
GPU: GeForce GTX TITAN X (Pascal)
Python version: 3.4.2
Libraries: tensorflow 1.0.0, numpy 1.12.0, argparse

Data Preprocessing

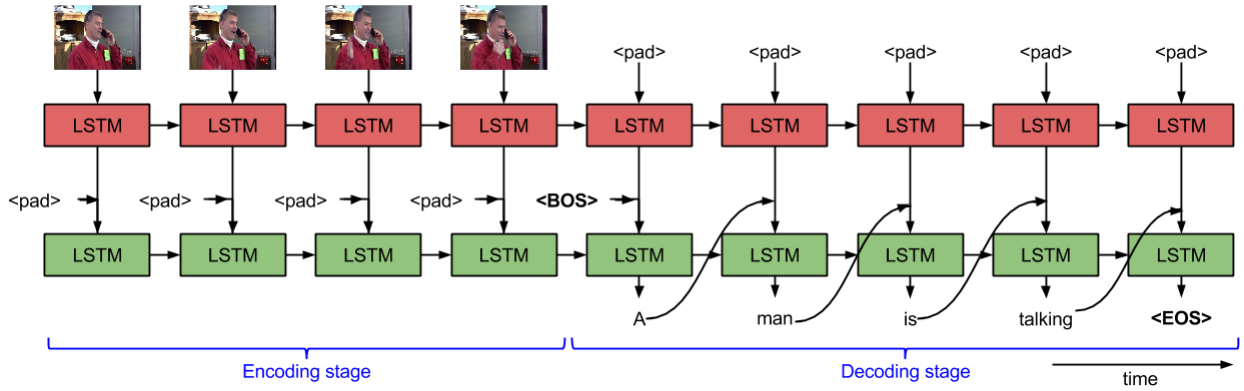
我們利用 training + testing data 內所有的 captions 建立字典 (vocabulary)，總計 6117 個字，包含下列四個保留字 <BOS>, <EOS>, <PAD>, <UNK>。由於字詞數量不大，所以沒有對字詞做篩選。

在 training 時，我們使用兩種不同的 caption 選取方式，從該影片的 candidate captions 中：**(a) 隨機挑出**一個 caption 當作答案、**(b) 挑出最長**的一個 caption 當作答案。不管使用 (a) 或 (b) 的選取方式，我們都會將挑選出的 caption 以 <PAD> 保留字將 caption 填補至預設的 caption長度，其長度取決於模型參數 n_caption_step，若 caption 超過設定長度，則該 caption 只截取至固定長度。另外，我們利用簡單的 mask 方法，將自行補上 <PAD> 造成的 loss 忽略不計。

Model

A. Basic Structure

我們依照 Sequence to Sequence: Video to Text 這篇 paper 提出的模型完成實作，model 架構如下：



在實作中，我們先將 video feature (4096 維/frame)投射至和 LSTM state_size維度相同的空間中，這樣一來，我們就可以在LSTM1 (red) 和 LSTM2 (green)補入相同維度的 <pad> ，並且按照上方 two stacked LSTM layers 的架構建立整個模型。

B. Schedule Sampling:

我們亦實作 schedule sampling 以解決 mismatch between training and testing 的問題。在我們的實作中， probability of sampling from model 會隨著 epoch 增加而漸增，目前設計為 initial sampling probability 乘上 n_epoch/50，舉例來說： 50 epochs 後是原本的 2 倍、100 epochs 後則是原本的 3 倍。越後期利用 predicted word at time t 當作 input at time t+1 的機會越高。

C. Attention-based Model

根據課堂上講解的 attention-based model 實作，步驟與程式說明如下:

Encoding stage

1. 記錄每個時間點 video_frame 輸入至 encoding stage LSTM1所輸出的 state。

Decoding stage

1. 使用初始 matching vector (z^0, trainable) 對每個時間點 t 的 encoding stage LSTM1所輸出的 state 計算 matching 分數 (α^t)。
2. 將 encoding stage LSTM1所輸出的 states 以 α 做 weighted sum，得到 c^0。
3. 將 c^0 輸入至 decoding stage LSTM2，輸出一個預測的 word。
4. LSTM2 state output 取代 z^0 成為新的 matching vector，重複 1~4。

```
# Encoding stage
for t in range(0,n_video_step):
      output1_t , state1_t = LSTM_1(video_frame_t)

      # record the LSTM_1_state every time
      state_history.append(state1_t)

# Decoding stage
for l in range(0,n_caption_step):
      # calculate matching
      α^t,l = h^t W z^l, where t=1,2,...,n_video_step

      # Sum up those LSTM_1_state(s) weighted by α
      c^l = Sum of α^t,l * h^t, where t=1,2,...,n_video_step

      # input c^l to LSTM_2
      output2_t, state2_t = LSTM_2(word_l,c^l)

      # replace the matching vector by LSTM_2_state
      z^l+1 = state2_t

      # output the word with the highest probability
      pred_word_l = argmax(output_de_t)

      # update the variable, word_l
      word_l = pred_word_l
```

Matching function

採用 $\alpha = H^T WZ$ 的型式，其中 W 和 z^0 是 trainable，每個變數的 dimension 如下：

H = n_video_step x LSTM_1_state_size

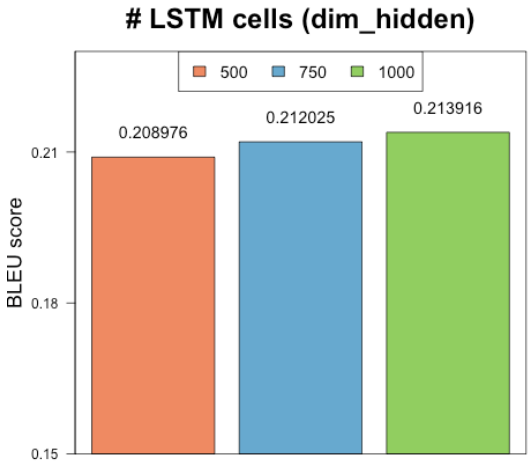
W = LSTM_1_state_size x LSTM_2_state_size

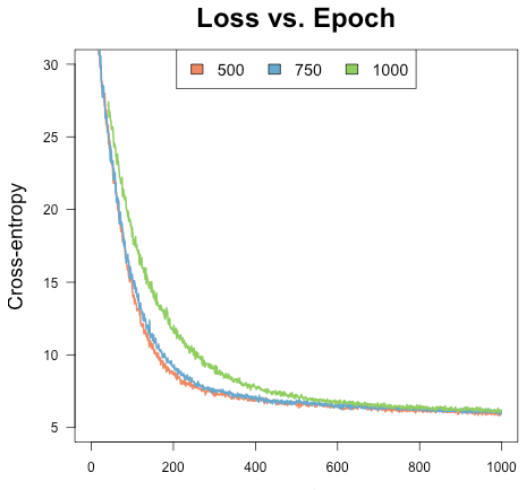
Z = LSTM_2_state_size x n_caption_step

Experiments

Exp0: 比較不同大小的 LSTM cells 對 performance 的影響。

其他參數則使用 default setting (please find the details on github)。從實驗結果發現，越大的 LSTM cells所得到的效果越好，但由於 GPU memory 限制，最後採用 dim_hidden=1000。





Exp1: 比較 (a) 隨機挑選一個 caption (b) 挑選最長的 caption 當作答案的差異。

相較於隨機挑選，訓練時挑選最長的 caption 當作答案，會使得 model 輸出的 caption 長度平均多 **7.38** 個字，是 best result 的兩倍之多。我們期待使用較多的字數會有較細緻的描述，但從 BLEU 來看，平均 BLEU score 為 **0.180591**。下面舉一個例子 testing_id 為 "BAf3LXFUaGs_28_38.avi"。

其中一個答案： a man is playing the drums while two woman play pianos

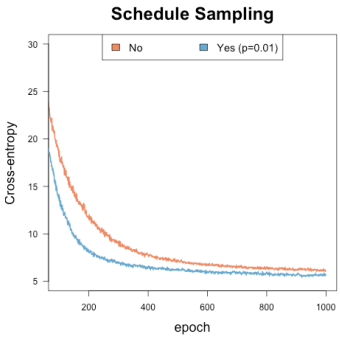
Model (a), BLEU score = 0.400 ⇒ a man is singing and playing the guitar
Model (b), BLEU score = 0.219 ⇒ a man is singing and playing the guitar on stage along with a band of musicians

用 BLEU 來評估的話，Model using (b) 輸出太多字反而得到較低的分數，但單看上面的範例，Model (b) 確實有掌握到更多而且正確的細節。在隨機挑選 captions 當作答案有近似 soft computing 的感覺，沒有一個制式的標準答案，反而讓 model 有更好的 generalization。

Exp2: 比較使用 schedule sampling 對 performance 的影響。

使用 schedule_sampling 的確可以減少 mismatch between training and testing，但是效果並不顯著。其他參數皆使用 default setting。

- With SS (p = 0.01), BLEU score = 0.217354
- Without SS ⇒ BLEU score = 0.213916



Exp3: 比較使用 attention-based model 對 performance 的影響。

由於 GPU memory 限制，我們設定attention-based model 的 dim_hidden = 200 , batch_size = 50。在此限制下，attention-based model 沒有辦法增進 performance。目前使用 attention-based model，最好的 BLEU score 為 0.190019。

Performance

- **Baseline:** 使用 S2VT without attention model 訓練 2000 個 epochs，所花時間為 4 小時。最終 average BLEU score 為 **0.275**。
- **Best:** same as Baseline model。

Team Division

f03942038 鍾佳豪	Model training;
r05942102 王冠驊	Debug; Model training; Report
d05921027 張鈞閔	Implementation of S2VT, schedule sampling, and attention; Model training; Report
d05921018 林家慶	Experiment design; Report