MLDS 2017 Spring HW2 - Seq2Seq & Attention

B03901056 孫凡耕 B03901070 羅啟心
B03901032 郭子生 B03901003 許晉嘉

1 Environment

OS	CPU	CPU Memory	GPU	GPU Memory
`Arch linux 4.10`	`i7 3.6 GHz`	`32 GB`	`GTX 1080 Ti`	`11 GB`

2 Model description

1. Seq2Seq 模型：
   典型的 encoder-decoder 的模型，也就是利用 LSTM 把不同長度的輸入，轉化為固定大小的 state，然後將此 state 傳遞給另個 LSTM，並依序輸出當前最佳解直到遇到 `<eos>` 爲止。
   模型所使用的參數如下：

   - batch_size = 145
   - video and word embedding dimension: 4096
   - layer_num = 2
   - rnn_cell = LSTM with peephole
   - hidden_size = 256
   - optimizer = adam
   - dropout keep prob = 0.7
   - with bidirectional rnn at encoder
   - with attention (bahdanau)
   - with scheduled sampling (linear decrease to 0.5)

2. S2VT 模型：
   在 sequence to sequence 之外，我們也根據 paper 實作出了 S2VT 的模型。不同於傳統 seq2seq 是由一層 encoder 和一層 decoder 所組成，S2VT 是由兩個 LSTM cell 在時間軸上分成 encoding 和 decoding 的階段。影片在 encoding 階段每個 frame 做 embedding 至 500 維，作為第一個 LSTM 的輸入，第一個 LSTM 的輸出和 `<pad>` 做 concat 之後即為二個 LSTM 在 encoding 階段的輸入；在 decoding 階段，第一個 LSTM 輸入不再是影片的 embedding，而改為輸入 `<pad>` ，第一個 LSTM 的輸出和 embedded 的 word 做 concat 之後即為第二個 LSTM 的輸入。值得一提的是 decoding 階段要以輸出 `<bos>` 作為開頭，並以第二個 LSTM 輸出的 `<eos>` 當作結尾。模型參數如下：

   - batch_size = 290
   - video and word embedding dimension: 500
   - rnn_cell = LSTM with peephole
   - hidden_size = 1000
   - optimizer = adam
   - dropout keep prob = 0.7

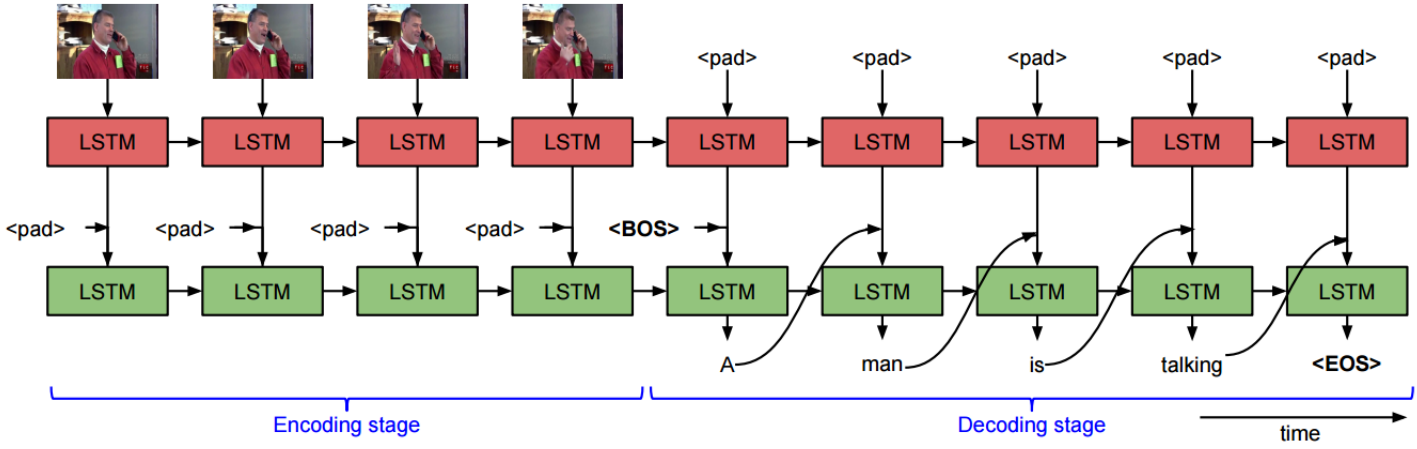

Figure 1: S2VT 模型架構

3 Improvement

我們在 seq2seq 的模型之中做了許多的 improvement，包含加上 attention、bidirectional、schedule sampling 等等。而在傳統的 seq2seq 模型之外，我們也根據 paper 嘗試使用 S2VT 作為本次作業的 improvement，並在 S2VT 上加了 schedule sampling 做更進一步的優化，以及一些不同於 paper 在參數上的調整實驗。比較可惜的是在時間內我們並沒有在 S2VT 的模型中看到比 seq2seq 更好的結果，而最終 S2VT 模型在 bleu 分數表現上最好的結果大約是 0.22 左右。一些輸出的結果如下：

- `a man is dumping pasta into a plate`
- `a man is playing the guitar on stage`
- `a person with a cleaver is slicing an onion`

1. Attention
   我們嘗試比較不同 Attention 以及使用 Attention 與否。從 Figure 2 中可以發現，當不使用 Attention 時，學習的過程十分緩慢，並且訓練的效果極差，幾乎無法有學習的效果。但加上 Attention 之後，學習的效果便十分顯著。而兩種 Attention 的方式大致來說並沒有顯著的差異。

2. Bi-directional
   我們嘗試比較使用 Bi-directional 與否。並分爲使用 Attention 與否的兩種情況討論：

   - 不使用 Attention
     從 Figure 3 中可以看出，在不使用 Attention 的情況下，加上 Bi-directional 時，學習的過程會比較緩慢，並且學習的效果並不會比較好。我們推論，這是由於不使用 Attention 時，由 Bi-directional 從後方 feedback 回來的資訊，對於當前的 cell 而言，就相當於雜訊一般。因此，不僅無法提升學習的效果，反而會阻礙學習的過程，因此，在訓練上的學習效果便會比較差。
   - 使用 Attention
     從 Figure 4 及 Figure 5 中可以看出，當使用 Attention 時，加上 Bi-directional 會加強學習的效果，這是由於有 Attention 的幫助下，能學習到 Bi-directional 回來的資訊以及前方資訊的關聯，進而提升訓練的成效。

3. Scheduled Sampling
   從 Figure 3 到 Figure 5 中可以看出，使用 Scheduled Sampling 會使 perplexity 的變化較爲動盪，但在這次的作業中，使用 Scheduled Sampling 並沒有顯著的提升學習成效的現象。

4. Word embedding
   原先的模型中，我們對於任意一個單詞，是先隨機初始值來進行訓練，但事實上，我們應該是有辦法先對 caption 進行分析，來達成更好的訓練效果。因此，我們將 caption 的單詞先蒐集起來，自己訓練出這些單詞的 word embedding，如此一來，就將變數改爲固定的 pretrained 值，而不是

隨機的變數。另外，我們原先使從若干句 caption 中挑一句出來作爲訓練資料，其他句子便完全沒有運用到。如此一來，我們所看過的句子就只有一千多句。而現在我們預先訓練自己的 word embedding，便能利用到其他 caption 所提供的語料資訊。

5. Black Magic
   原先的訓練資料是以 80 個 frame 的影片對應若干個 caption，而我們原先是將一個影片對應一個 caption 做訓練。而現在，我們可以將 80 個 frame 分爲五組不相交的 16 個 frame 分別對應一個 caption，相對於分爲五個影片。由於訓練模型的結構，我們同樣可以以這組資料做訓練。而測試的時候，同樣以 80 個 frame 作爲輸入，如此訓練出來的效果，會比原先的結果來的較好。原先的 bleu score 大約 0.29，使用此方法後，bleu score 最高進步到 0.32。

6. from static to dynamic
   原始的 S2VT 我們使用 for 迴圈進行實作，然而這種做法會使得 model 在 tensorflow 中被展開使得記憶體使用量過大，因此我們最後 S2VT 是用 tensorflow 當中的 raw_rnn 搭配 tensorflow 內建的 loop function 實作，如此便可避免記憶體使用量的問題。

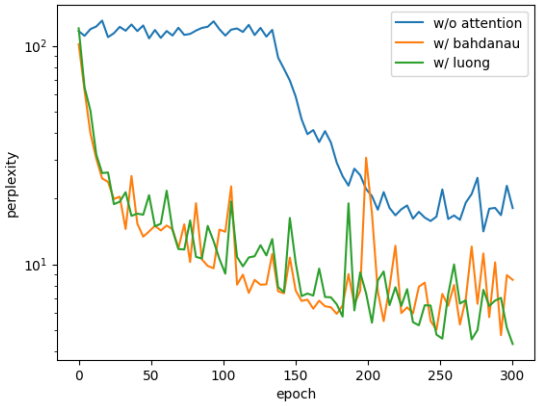

Figure 2: 不同 Attention 間之比較

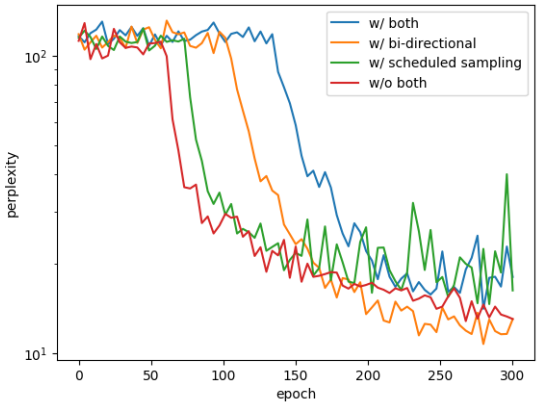

Figure 3: 不使用 Attention

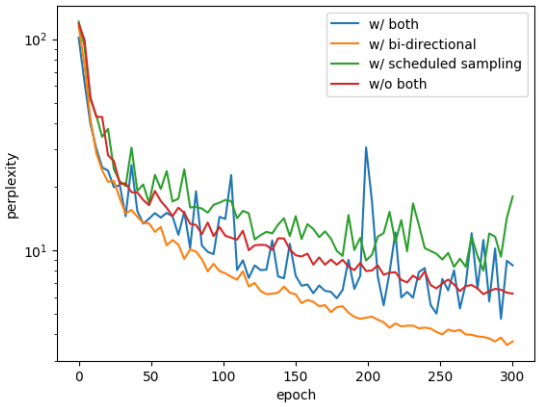

Figure 4: bahdanau Attention

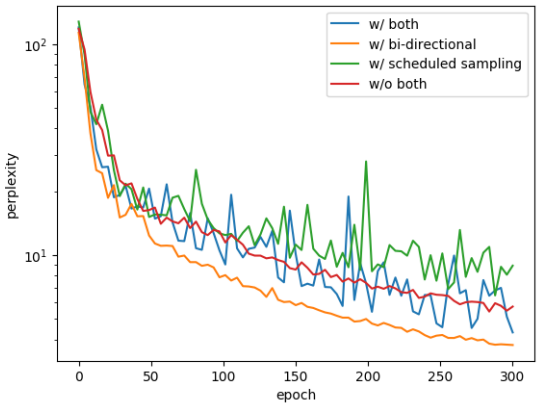

Figure 5: luong Attention

4 Experiment

1. 對於 S2VT 模型，我們首先對於原始 Paper 的 Model 進行重現，接著便在參數以及模型上進行嘗試與調整。在 Cell 種類的改變下，如 Figure 6 所示，可以發現在 training 的過程中的確有 GRU、快於 BasicLSTM、又快於 FullLSTM 的趨勢，但整體表現上還是 FullLSTM（含有 peephole ）的表現較佳。另外我們對於每個影片只取出一個 caption 為該影片中所有 caption 長度中位數的做 training，並從影片的 80 個 frame 中每 1, 4, 8, 10 的 frame 取一張做為 input，可以發現當取出一個 caption 以及每 4 個 frame 取出一張會在 bleu 分數上稍有進步，但並沒有太大的不同。此外我們也在 S2VT 的模型中加入 schedule sampling，但 bleu 的分數反而下降到 0.194 左右，並沒有比原先的 S2VT 更為進步。

2. 從 Figure 7 中可以看出，在剛開始的時候（大約 200 epoch 時），bleu score 就已經很高，超過 baseline 達到 0.28 左右，但事實上，每次所預測出的句子都是 `A man is playing a guitar`，所以實際上並沒有學習的效果。雖然隨着 perplexity 的下降，bleu score 也跟着下降，但實際上所學習出來的句子更爲精準。但因爲模型所預測出的句子，有時候無法切題，因此 bleu score 相對來說，反而略微下降。當 perplexity 掉到 5 以下時，此現象更爲明顯。例如，會把 man 預測成 woman。

3. 因爲我們 vocabulary size 只有 6000 左右，所以 perplexity 要掉到 5 以下，每個句子的預測才會有所不同，如下表 300 個 epoch 時，所有的預測都是 a man is playing a guitar，當 perplexity 下降到 $2 \sim 3$ 左右時，預測出的句子有更高的多樣性，但當 perplexity 繼續往下掉到 1 時，許多句子反而會預測出錯誤的結果，如下表右下的例子。可見，perplexity 和預測的結果有一定的一致性，但 perplexity 過低反而代表出現 overfitting。

4. 對於不同的 embedding size，雖然在 1000 個 epoch 以後四種 embedding size 的 perplexity 都會降至 50 左右，但是由 Figure 8 中可以看出，四種 embedding size 的 perplexity 都會停留在約 160 一段時間，embedding size 為 4096 的 perplexity 大約在 200 左右就會繼續下降，而 embedding size 為 2048 的 perplexity 大約在 240 左右才會繼續下降，embedding size 為 1024 和 512 則在 perplexity 為 160 左右停留更久，分別是 420 和 700 個 epoch 之後才有下降的趨勢.

epoch	perplexity	score		
300	8.74	0.291	`a man is playing a guitar`	`a man is playing a guitar`
600	3.03	0.272	`a man is riding a horse`	`a man is walking on a treadmill`
900	1.84	0.245	`a man is riding a horse`	`a man is swinging on motor bike`
1200	1.43	0.248	`a man is riding a horse`	`a man is swinging on a bulldog`
			`a girl is jumping rope`	`a man is playing guitar`
epoch	pexplexity	score		
300	8.74	0.291	`a man is playing a guitar`	`a man is playing a guitar`
600	3.03	0.272	`a man is pouring plates to a car`	`a man is playing a guitar`
900	1.84	0.245	`a man is firing a gun`	`a man is playing the drums`
1200	1.43	0.248	`a man is firing a ukulele`	`a man is playing a trumpet`
			`a man is screaming`	`a man is playing drums`

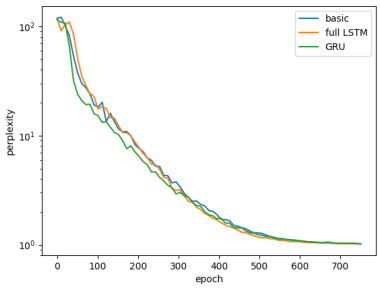

Figure 6: S2VT 結果

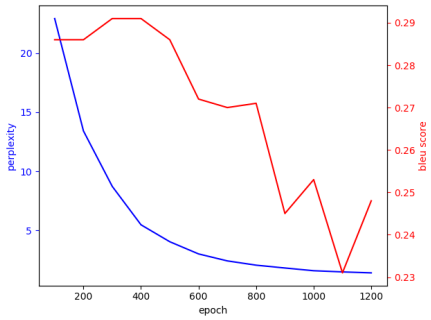

Figure 7: preplexity 與 score 的關聯

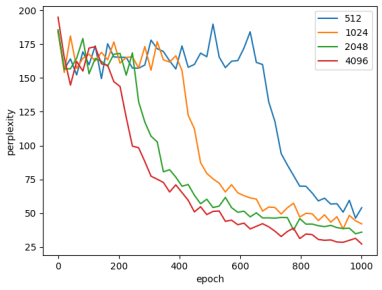

Figure 8: 不同 embedding size

5 Team division

孫凡耕	Seq2Seq、分配組內工作、教導組員
羅啟心	Tensorboard、Experiment
郭子生	S2VT、Experiment
許晉嘉	統整撰寫報告、跑實驗、優化 word embedding