MLDS -- HW4 Seq2seq & RL -- Report

B03202017 李漪莛，B03202047 王昱翔

1 Environment & Data Sets

OS	Ubuntu 16.04.2 LTS	Memory	4G
CPU	Xeon E5-2630V3	GPU	NVIDIA GTX 970
Libraries	Tensorflow r1.0, numpy 1.12.0, CUDA8.0		
Data Set	Cornell Movie-Dialogue Corpus		

2 Model Description & Reward Function

2.1 預處理

- Data set 整理成前後兩句為一組對話：input & response。
- 字母轉成小寫，數字轉成 “Number”，每句頭尾加上 BOS、EOS。
- Vocabulary size = 10000，input 中沒見過的字轉成 Unknown，response 中沒見過的字刪掉，若要刪掉連續兩個字，則把整組對話刪掉。（原因在如 3.5）
- Input 取後 15 個字，response 取前 15 個字，不夠長則補零，

2.2 Seq2Seq

- Input (15*10000) → FC (fully-connected) → 15*512 → encoding LSTM:tf.nn.dynamic_rnn (BasicLSTMCell) → 取出最後一個 num_steps 的 output，令為 hidden。
- Response (15*10000) → FC → 15*512，令成 last_word。
- 把前一個 num_steps decoding LSTM 輸出的字（1*512）直接取 argmax（1*1）→ one_hot vector（1*512），令成 last_word。
- 一一取出上兩個步驟的 last_words(Schedule Sampling)：Concat[hidden, last_word] → 1*1024 → FC → ReLU →1*512 → decoding LSTM → 取出 output → FC → ReLU → 1*10000 → argmax → output words。

2.3 Reward Function

- R1：p = model 輸出對應到 dull set 中的字的機率，R1 = -mean1(mean2(log(p)))。
  意義：讓 model 避免輸出 dull 句子。
  我們定義的 dull 的句子有五句，如下：I don't know., I have to go., no, I don’t., no, I dont know what you are talking about., I dont want to see you.
- R2：p = model 輸出的字對應到 EOS 的機率，R2 = -mean1(mean2(log(p)))。
  意義：少輸出 EOS 增加句子的長度。
- R3：p = 輸出的字對應到 data set 中正確的 response 中的字機率，R3 = mean1(mean2(log(p)))。
  意義：增加回答正確的字的機率，與一般 seq2seq 的目標函數類似。

- 以上的 mean1 指對一個 batch 中所有句子的分數做平均，mean2 指對一個句子中的字數做平均。
- R = c1*(R1-b1) + c2*(R2-b2) + c3*(R3-b3)，ci 的值記錄在下面，(b1, b2, b3) = (10, 40, -3)。
- 直接 maximize R。

3 How to improve performance

3.1 增加 Seq2Seq LSTM units 數量

對於 Seq2Seq，encoding 和 decoding 的 units 數目都定為 256 時，正確率約為 33%，改成 512 後，正確率可到 36~37%。

3.2 增加 Embedding 維度

當 embedding 維度是 128 維，有八成的回答是 I don't know.，當我們將 embedding 維度和 units 數目都改成 512 時，只有兩成的回答是 I don't know。

3.3 調整各種 Reward 的比重

訓練好 Seq2Seq 後，先分別針對我們定義的 R1、R2、R3 做優化，觀察三者的成效如何，方便我們知道怎麼調整到合理的訓練比例（c1~c3），這部分是 RL 中使我們進步最多的方法，因為一旦比例不對，很容易造成文法特性喪失。（相關結果在 4.2）

3.4 不再訓練 Seq2Seq

我們原先以為同時優化 RL 和 Seq2Seq 比較能保值文法特性，但實際上，會造成出現許多不通順的句子，例如：i Unknown . i . EOS . EOS .；因此我們把 Seq2Seq 訓練好後就不再繼續優化。

3.5 去掉 Response 的 Unknown

原先，在訓練資料的 response 中若出現沒在 vocabulary 中的字，就把它 token 成 Unknown，但這會使 testing 的回覆句子中出現很多 Unknown，導致 post-process 時如果直接把 Unknown 拿掉，句子會很不通順。因此我們在預處理時，把 response 的 Unknown 去掉（Input message 中的繼續留著），直接銜接下一個字，若 Unknown 太多就不把這組對話當作訓練資料，可使 testing's response 的句子更通順。

3.6 不用 Attention

我們發現加入 attention 後回覆的句型並沒有明顯的差別，還導致整體訓練速度緩慢許多，因此我們在後續的訓練過程不採用 attention。

4 Experiments settings and observation

4.1 Settings

- Learning rate = 0.001~0.0001
- Epoch：Seq2Seq = 30, RL = 0.5
- Batch size = 64
- Optimizer = Adam

4.2 各種 Rewards 的結果整理

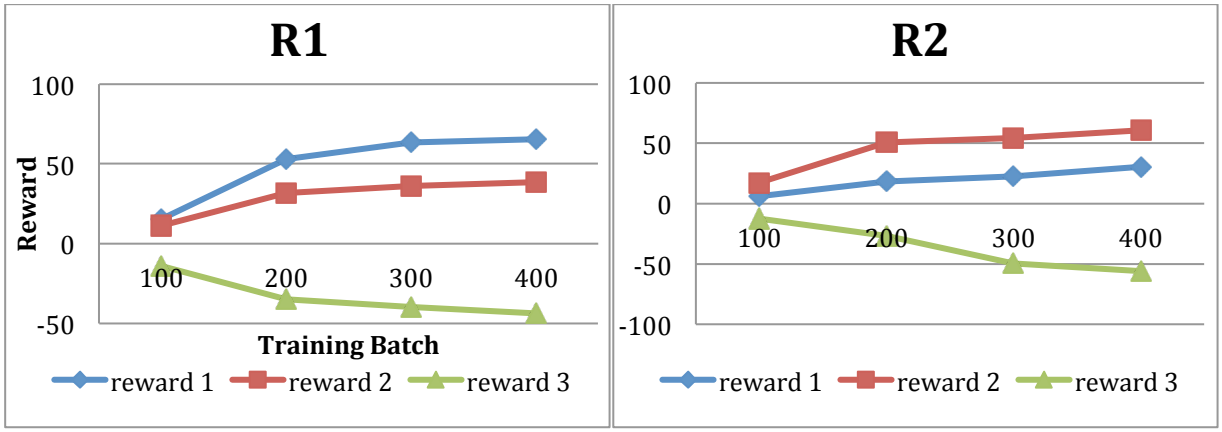
- 輸入相同的 4 句 input message，比較 response 的差別。
- 所有情況最好的結果，以「good」表示。
- 若 RL 訓練過久，可能喪失文法，我們記錄這筆結果，以「bad」表示。

	Resp 1	Resp 2	Resp 3	Resp 4
Input	what is your name ?	are you a boy ?	hello , how are you ?	how to go to supermarket?
Seq2Seq	.	yes .	fine .	I dont know .
R1 - good	what are you talking about ?	yes .	fine .	I dont know .
R1 - bad	（都是） you lose .			
R2 - good	I dont know .	yes .	fine . ill be right there .	I dont know .
R2 - bad	（都是）i dont know . i dont know . i dont know .			
R3 - good	im not sure .	yes .	fine .	i dont know .
R3 - bad	yeah .	yes . it is .	fine .	i dont know .
Final - good	.	yes , sir .	fine .	i dont know .
Final - bad	yeah .	a .	fine .	you not the one day ?

- 只優化 R1 的結果

  第 200 個 batch reward 上升很快，最好的結果也是這時候得到。然而之後訓練的回覆開始不合文法，且繼續訓練下去，回覆都不再改變了。 此外，可看出只優化 R1 時，R2 也會跟著增加，R3 卻退步得很嚴重。（和下一點一起討論）
- 只優化 R2 的結果

  R2 以句子長度做評分，出現 EOS 就扣分。在 testing 中，第 200 個 batch 以後的回答都是 i dont know . i dont know . i dont know . （連續三次）。我們認為是 R2 單方面降低 EOS 的機率，就會直接輸出原先機率最高的句子：i dont know .，這也是他最省力的方法。 觀察到只優化 R2 時，R1 也跟著進步，但 R3 仍會下降。原因可能是：自定義 dull 句子字數都偏短，R1 在產生非 dull 句子的過程中，也更有機會產生較長的句子，所以 R2 會跟著提高。而 R2 產生這些較長的句子都並不在 dull sentences 中，所以 R1 也會跟著增加。然而不管 R1 或 R2，更新方向都與 R3 不同，才導致 R3 下降。



- 只優化 R3 的結果

  可觀察到 R3 跟其他 reward 相比，優化幅度非常小。

  我們推測原因是：R3 是答對時加分，而 R1、R2 是答錯時扣分，因此 R3 比較難訓練。並且，R1、R2 的分數不需考慮 input 的內容，只要看到 dull 句子或句子太短就扣分，R3 還需要考慮 input 的不同來調整 response 的方向。

- 同時訓練 R1, R3 的結果

  從結果上看 R2 效果不大，所以我們最終 reward = 0.1*R1 + 0.05*R2 + 0.85*R3。

4.3 不同 Model：SeqGAN

我們嘗試加入 discriminator（簡稱 D）來當作新的 reward，D 的架構參考：https://arxiv.org/pdf/1609.05473.pdf。Label1 表示正確的回覆，Label0 表示產生的回覆。我們將先原本的 RL，和 SeqGAN 等比例輪流訓練。然而我們發現 D 的 loss 很快就會下降到很低，但回覆反而變得更差。

D_loss 很低的原因可能是產生的句子只要文法不對，或一直產生通用句型，都會被 SeqGAN 辨認出來。所以要訓練 SeqGAN 的 model 可能要更複雜一點。

5 Team divisions

李漪莛：preprocess、Seq2Seq、RL、report

王昱翔：SeqGAN、post-process、report、彙整上傳