MLDS HW1 Language Model Report

b03202017 李漪莛，b03202047 王昱翔

1 Environment

- OS: Ubuntu 16.04.2 LTS
- CPU: Xeon E5-2630V3
- GPU: NVIDIA GTX 970
- Memory: 4G
- Libraries: Tensorflow r1.0, numpy 1.12.0

2 Model description

我們最好的 model 跟 RNN 的 model 是一樣的，所以下面都只敘述一個 model.

2.1 預處理

- 從句子是“*END*THE SMALL PRINT!”開頭的往後十句，開始納入訓練文本 corpus，直到某句子開頭是“End of Project Guten”或“End of The Project G”的前十句以後，不再納入 corpus。
- 若某句子開頭是任意大小寫的“chapter”、“part”、“句子是數字開頭”等小標題，則移除此句子。
- 若在句中遇到“。”、“？”、“！”、對話等，則把句子分開。
- 移除句中的“，”、“：”、“；”等標點符號。
- 把 500 多本小說中剩下的句子串接起來定為 corpus，計算最常見的 11000 個單字，視為 common_word_list，再把 testing_data.csv 中的所有選項，移除雙字詞後，剩下的詞也加入 common_word_list。其他不在 common_word_list 中的字則定為 unknown_word，並把 corpus 中句子長度少於（含）四個字元的句子去掉，最後 corpus 共得到 180 多萬句。

2.2 處理 testing_data 的選項問題

testing_data 中總共有 1040 道題目，每題五個選項，因此約有 5000 個選項。扣除重複出現的選項後轉成小寫，若此選項沒有“－”，則把此選項加入“choice_list”。

其他含有“－”的選項：將它們一一和 common_word_list 中的 11000 個字比對，若去掉“－”後出現在 common_word_list 中，則把此選項加入 choice_list（如：to-morrow 轉為 tomorrow），若沒有，則把這個選項剔除（如：double-edge 等雙字詞），此步驟約剔除掉 80 個左右的雙字詞，影響應不至於太大，做完此步

驟後，choice_list 中的詞總共有 3591 個。

2.3 LSTM model

- 輸入：11792 個詞的 one hot vector。
- 經過 11792*800 的矩陣 w1 和 bias（b1）轉換成 800 維。
- 兩層 800 個神經元的 LSTM（tf.contrib.rnn.BasicLSTMCell in tensorflow），預設 forget_bias = 1，訓練時 dropout = 0.4。
- 經過 800*3591 的矩陣 w2 和 bias（b2）轉換成 3591 維輸出。
- 輸出：利用 tf.nn.softmax_cross_entropy_with_logits()計算 loss function。

3 How do you improve your performance?

- 訓練樣本--1：由於 vocabulary size 很大，若只取 12000 個常見的字，把這個題目當作分類問題來看，選項還是很多，因此我將輸出（output layer）調整成 3591 個，只會輸出 testing_data 中選項中可能出現的字，並且，在 corpus 中只把含有 choice_list 裡面的詞的句子才能出來訓練 model，可更準確的訓練。共計找出 63.6 萬句，其中約 1750 個詞，出現 260 次，其他詞由於在 corpus 出現頻率更低，平均約出現 100 次，不過，透過這種做法，能讓每種“class”的訓練樣本盡量平均。
- 訓練樣本--2：把 rnn 可容納句子的長度從 20 增加為 30 後，訓練時也比較容易收斂。
- Shuffling：這一點滿重要的，且我在兩個地方有做 shuffling。
  第一，因為我有切六千筆資料作為 validation，並且我的訓練樣本不同部位的 domain 不太一樣，因此在切出 validation data 之前，一定要先做過 shuffling。
  第二，每跑完一個 epoch 也要做 shuffling，這可能跟訓練順序有關，收斂方向會傾向比較早被訓練到的那一群，就算比較晚被訓練到的資料多訓練幾次，仍無法改善。
- Layer 選擇：我嘗試在兩層的 lstm neuron 的前、中、後各自加上一層普通的 layer，包含沒有 acitvation function、使用 reLU 或 sigmoid，結果發現都沒有比純兩層 lstm 來得好（要多花更多時間，並且 training acc 很難超過 10%）。
- 防止過擬和：由於樣本數只有 63 萬句左右，很容易 overfitting，因此加上 dropout rate= 0.4。
  且一開始考慮使用較簡單的 model，如：只有一層 lstm，或 lstm 單元數較

少，不過發現較複雜的 model 雖然 overfitting 較嚴重，但可以在 validation loss 較小（收斂）之後，才開始 overfitting，並且收斂速度快很多，因此仍選擇稍微複雜點的 model。

- L2 項：一開始嘗試在 loss function 加上 L2 項，但反覆調整 L2 的常數項後，發現不管是 training 或 validation 的結果都沒有比較好，因此後來就拿掉這一項。

4 Experiment settings and results

- Optimizer：AdamOptimizer
- Epoch：約 5/3 個（共跑 5250 個 batch），經過測試後，此時 validation loss 較低。
- Training Time：一個 epoch 約 45 分鐘，因此重現 model 約 75 分鐘。
- Learning rate：第一個 epoch -- 0.001，其餘 -- 0.0003
- Dropout：0.4 in training、1（no dropout）in validation and testing
- Batch_size：200
- Training：accuracy 再經過 4、5 個 epoch 普遍都能輕易達到 80~90%的正確率。
- Validation：目前測試最好的 model 也只能在 validation 達到 6~7%的正確率，不過由於 Kaggle 上的答案可五選一，因此這個 model 恰可過 public baseline，正確率約 31.9%。

  若用較小的 corpus（如 40~60 萬句子不等）或較簡單的 model，會得到更低的 validation 正確率。一般來說，在 training 正確率為 0.18~0.25 左右會有最高的 validation 正確率和最低的 validation loss。
- Public score: 0.319, Private score: 0.336

5 Team division

李漪莛：預處理、建 model、打報告

王昱翔：建 model、打報告、整理上傳程式碼