

ADL HW1 Report

R10922124 林家毅

Q1

Intent Classification

- a. 因為在 data (train.json & valid.json) 裡面的 `text` 欄位都是一個完整的句子，所以在前處理的時候先把每個句子用空格拆開，紀錄每個 word 出現次數由大到小排序，並取最常出現的 10000 個字，未來看到這些 word 的時候用 index 對應；
`intent` 欄位則是代表某個類別，所以利用一個 set 收集所有 unique 的類別，未來看到這些類別也可以用 index 對應。
- b. embedding 的部分是利用 glove.840B.300d.txt，將 glove 裡面每個出現在 data 裡面的字取出來，對應該字的 index 到 300 維向量，如果有一個 data 裡的字在 glove 裡面沒有出現過的話就賦予它一個隨機 300 維向量

Slot Tagging

- a. 因為 data (train.json & valid.json) 裡面的 `tokens` 欄位已經是一個字串陣列，所以在前處理的時候直接算出每個 word 出現次數由大到小排序，並取最常出現的 10000 個字，未來看到這些 word 的時候用 index 對應；`tags` 欄位則是類別陣列，一樣是利用一個 set 收集所有 unique 的類別，未來看到這些類別也可以用 index 對應。
- b. embedding 的部分跟 intent classification 的方法一樣，都是利用 glove.840B.300d.txt，將 glove 裡面每個出現在 data 裡面的字取出來，對應該字的 index 到 300 維向量，如果有一個 data 裡的字在 glove 裡面沒有出現過的話就賦予它一個隨機 300 維向量

Q2

- a. model 是由 1 layer Bidirectional GRU 以及 1 layer fully-connected output layer 所組成，其中 GRU hidden dimension = 128，不過因為是 bidirectional 所以 fully-connected layer 的 input 會是 256 維，輸出則是 150 維 (number of classes = 150)

- b. Public Score: 0.91466
- c. Loss Function: Cross Entropy Loss
- d. Optimization Algorithm: Adam with learning rate = 0.001 and batch size = 16

Q3

1. model 是由 1 layer Bidirectional LSTM、10 層 fully-connected hidden layers 以及 1 層 fully-connected output layer 所組成，其中 LSTM hidden dimension = 128，而所有的 hidden layers 都是 256 * 256 加上一層 relu 和一層 dropout (= 0.35)，最後的 output layer 則是 256 * 9 (number of classes = 9)
2. Public Score: 0.75227
3. Loss Function: Cross Entropy Loss
4. Optimization Algorithm: AdamW with learning rate = 0.001 and batch size = 64

Q4

- eval.json 中共有 1000 筆 data，分別計算 Joint Accuracy, Token Accuracy 以及利用 `seqeval` 輸出 Classification Report 的結果如圖一
- 從圖中可以發現，Token Accuracy 比 Joint Accuracy 高出許多，這是因為 Joint Accuracy 要求一個句子所有 tags 都預測正確才算正確，而 Token Accuracy 只需要該 token 的 tag 預測正確。而在 Classification Report 中因為設定 scheme = `IOB2`，所以會依照 Joint Accuracy 的方式，只有當 chunk 裡面的所有 tags 都被預測正確時才算正確

```

Joint Accuracy: 0.763 ( 763 / 1000 )
Token Accuracy: 0.960 ( 7578 / 7891 )
Classification Report:

```

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| date | 0.76 | 0.71 | 0.73 | 206 |
| first_name | 0.86 | 0.85 | 0.86 | 102 |
| last_name | 0.64 | 0.73 | 0.68 | 78 |
| people | 0.68 | 0.67 | 0.68 | 238 |
| time | 0.81 | 0.85 | 0.83 | 218 |
| micro avg | 0.75 | 0.75 | 0.75 | 842 |
| macro avg | 0.75 | 0.76 | 0.76 | 842 |
| weighted avg | 0.75 | 0.75 | 0.75 | 842 |

Figure 1: Joint Accuracy, Token Accuracy and Classification Report

Q5

因為 Q2 的 model 看起來有 overfitting 的問題，因此把 model 的 GRU 改成了 LSTM 試試看，結果發現在 train set 上兩個方法都 fit 的很好，如圖二；而在 valid set 上，LSTM 的 loss 明顯比 GRU 更低，overfitting 的問題看起來被減少了，如圖三 (藍色線是 GRU，橘色線是 LSTM)。

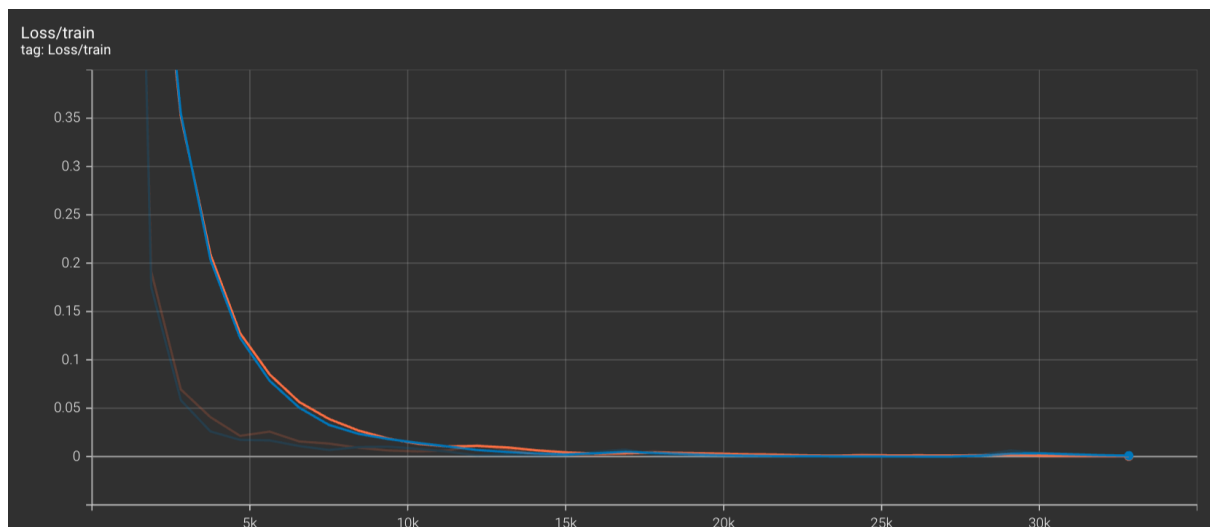


Figure 2: Training Loss

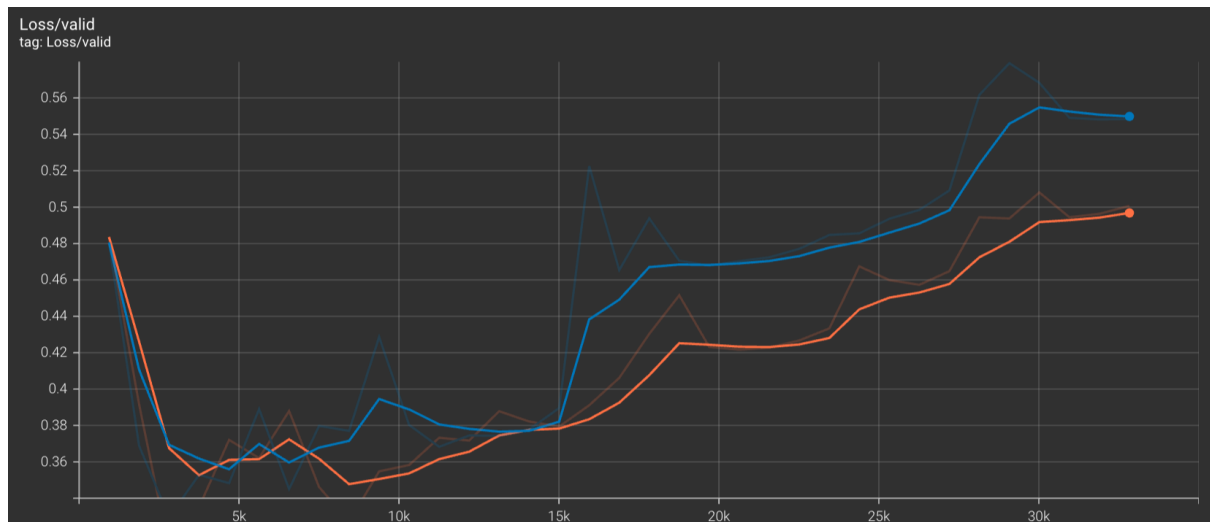


Figure 3: Validating Loss

Reference

1. [PyTorch Documentation](#)
2. [PyTorch 中的循環神經網絡模塊](#)
3. [pack_padded_sequence 和 pad_packed_sequence](#)
4. [SEQUENCE MODELS AND LONG SHORT-TERM MEMORY NETWORKS](#)
5. [Build Your First Text Classification model using PyTorch](#)
6. [LSTM Text Classification Using Pytorch](#)
7. [Sequence Tagging With an RNN](#)
8. [序列標註算法評估模塊segeval 的使用](#)
9. [Pytorch crossentropy loss with 3d input](#)