ADL HW1

r11922196 林佑鑫

Q1: Data processing

Describe how do you use the data for intent_cls.sh, slot_tag.sh:

a. How do I tokenize the data:

Step1. 對 intent 的 class 定出對應的 index，將對應關係寫到 intent2idx.json 中，對 slot 的 tag 同理，建立 tag2idx.json。

Step2. 紀錄 intent train.json 及 eval.json text 中的出現的所有單字，將 most common 的字建成 vocab 存到 vocab.pkl，對 slot 的 train.json 及 eval.json 同理。

Step3. 用 pre-train embedding 把 vocab 對應的 vector 輸出至 embedding .pt。

b. The pre-trained embedding I used:

glove.840B.300d
(用 840B 個單字訓練，裡面包含 2.2M 個單字，每個字用 300 dimensions vector 表示)

Q2: Describe your intent classification model

a. My model:

max_len	hidden_size	num_layer	dropout
150	256	2	0.1
bidirectional	**lr**	**batch_size**	**num_epoch**
True	1e-3	64	150

LL: largest text length in current batch, B: batch_size, L: max_len,
H: hidden_size, E: embedding dimension, C: num_class
因爲我的 L 設定很高，我假設 text 長度都不會超過 L，不會截斷，所以後面都是用 LL 來表示
這邊都用一個 batch 來看，若只看一個 sample 進 model 的過程可忽略 dimension 1。

Embedding layer: x(B, *) -> embed_x (B, *, E) ，* 是指每個 sample text 的長度不同
Padding: embed_x (B, *, E) -> packed_x(B, LL, E)
兩層 biLSTM: packed_x(B, LL, E) -> hidden(B, LL, 2*H)，兩層之間有 dropout layer。

fully connected layer: hidden(B, LL, 2*H) -> pred(B, C)
最後直接用 torch.max(pred, 1)[1]得到 predict 分數最高的 class index

b. Performance of my model:
(Public: 0.92577, Private: 0.91955)
這個 model 的 checkpoint 不小心被我洗掉了，而且我 shuffle 沒有固定 seed，沒辦法再 train 出一次，所以 reproduce 那邊交的是我做第二好的 model
(Public: 0.92177, Private: 0.92008)，在 README 中也有提到。

c. The loss function I used:
CrossEntropyLoss

d. The optimization algorithm, learning rate and batch size:
Optimizer: AdamW
Learning rate: 1e-3
Batch size: 64

Q3: Describe your slot tagging model

a. My model:

max_len	hidden_size	num_layer	dropout
256	1500	3	0.2
bidirectional	**lr**	**batch_size**	**num_epoch**
True	1.5e-3	64	100

LL: largest token length in current batch, B: batch_size, L: max_len,
H: hidden_size, E: embedding dimension, C: num_class
因爲我的 L 設定很高，我假設 text 長度都不會超過 L，不會截斷，所以後面都是用 LL 來表示
這邊都用一個 batch 來看，若只看一個 sample 進 model 的過程可忽略 dimension 1
Embedding layer: x(B, *) -> embed_x (B, *, E) ，* 是指每個 sample text 的長度不同
Padding: embed_x (B, *, E) -> packed_x(B, LL, E)
三層 biLSTM: packed_x(B, LL, E) -> hidden(B, LL, 2*H)，三層之間有兩層 dropout layer。
fully connected layer: hidden(B, LL, 2*H) -> pred_score(B, LL, C)
這邊本來要加上 CRF，但 loss function 一直無法搞定，最後沒使用。
最後直接用 torch.max(pred_score, 2)[1]得到 predict 分數最高的 tag index

b. Performance of my model:

(Public: 0.78337, Private: 0.78403)

c. The loss function I used:

CrossEntropyLoss

d. The optimization algorithm (e.g. Adam), learning rate and batch size:

Optimizer: AdamW

Learning rate: 1.5e-3

Batch size: 64

Q4: Sequence Tagging Evaluation

a.

```
              precision    recall  f1-score   support

        date       0.79      0.78      0.78       206
  first_name       0.91      0.89      0.90       102
   last_name       0.86      0.78      0.82        78
      people       0.71      0.70      0.70       238
        time       0.89      0.89      0.89       218

   micro avg       0.81      0.80      0.80       842
   macro avg       0.83      0.81      0.82       842
weighted avg       0.81      0.80      0.80       842

Train loss: 0.00 - joint_acc: 1.00 - token_acc: 1.00. Validation loss: 0.29 - joint_acc: 0.81 - token_acc: 0.97.
```

b.

Token accuracy: 正確的 tag 數 / 總 tag 數

Joint accuracy: 整句 tag 都對的句子 / 總句子數

Seqeval:

		Predicted condition	
	Total population = P + N	**Positive (PP)**	**Negative (PN)**
Actual condition	**Positive (P)**	**True positive (TP)**, hit	**False negative (FN)**, type II error, miss, underestimation
	Negative (N)	**False positive (FP)**, type I error, false alarm, overestimation	**True negative (TN)**, correct rejection

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$f1 - score = 2 * \frac{precision * recall}{precision + recall}$$

Precision: 預測爲正的樣本有幾%預測正確。
Recall: 眞實爲正的樣本有幾%預測正確。
f1-score: precision 與 recall 的調和平均，能同時呈現兩者特性。
macro average、micro average、weighted average 則是將所有類別結果綜合起來的指標。如果在每個類別總數不平衡情況下要將所有類別平等視爲一樣權重,適合使用 macro average; 要依每個類別總數佔比視爲權重,則適合 weighted average。如果在每個類別總數平衡情況下,就選擇 micro average。

Q5: Compare with different configurations

Intent classification:

1. 將 num_epoch 從 100 調高到 150，準確率有所提升，再調高到 200 時表現反而變差，應該是 overfitting 導致。
2. 將 batch_size 從 128 調降到 64 能提升準確率，應該是因爲 batch_size 128 訓練時 loss 太大，導致 model 在 train 的時候不如預期。
3. 將 max_len 調大準確率有所提升，可能是原本的 max_len 太小，使得一些句子被截斷，使得他們在 train 的時候無法呈現完整的特徵。

Slot tagging:

1. 將 hidden size 調高，從原本 128 調到 1000 再到 1500，準確率都有所提升，應該是因爲後面輸入的字有一定程度受前面單字的影響，保留更多之前的特徵時就能提高準確率，我後來還想再繼續調高，但 memory 已經用光，1500 是極限。
2. 將 num_epoch 從 100 調高到 150 時，表現不好，推測是已經 overfitting，故之後又減少到 80，一樣沒有很好，最後固定在 100。
3. 試過不同層數的 biLSTM，發現 2 layers, 4 layers 都沒有 3 layers 表現來的好。
4. 有試著實作 CRF，但 loss function 一直無法搞定，最後沒有使用。