魏廷儒 r10521601
CSIE : Applied Deep Learning
October 15, 2022

ADL HW1

Question 1. Data processing

1) intent 分類任務

   在 intent 分類任務上資料前處理部份，我是先將下載 glove 的 pretrained model，並且先將 train.json 及 eval.json 的"text" 資料分割成一個個單字 (總計有 6491 個不同的單字)，並看看 glove 是否有對應的單字的 embedding feature，如果有了話將其置換爲 300 維的 feature，如果沒有了話用亂數去取代。(其中有 5435 種單字有對應對的 embedding) 此外，我們也將 train.json 及 eval.json 的"intent" 的資料 encoding 成 0-149 的整數 (總計 150 個種類)。

2) slot tagging 分類任務

   在 slot tagging 分類任務上資料前處理部份，我是先將下載 glove 的 pretrained model，並且先將 train.json 及 eval.json 的"text" 資料分割成一個個單字 (總計有 4117 個不同的單字)，並看看 glove 是否有對應的單字的 embedding feature，如果有了話將其置換爲 300 維的 feature，如果沒有了話用亂數去取代。(其中有 3000 種單字有對應對的 embedding) 此外，我們也將 train.json 及 eval.json 的"tags" 的資料 encoding 成 0-8 的整數 (總計 9 個種類)。

Question 2. Describe your intent classification model.

1) Describe your model

   我將每個 batch 的資料先過一層 dropout layer(p=0.5)，之後接一個 GRU(hidden size=512, num layers=2, dropout=0.5)，我們取出最後一個 token 的 output，再接一個 linear layer 去預測分類結果。

$$w_t = DROPOUT(w_t)$$
$$output, h_t = GRU(w_t, h_{t-1})$$
$$cls = LINEAR(output[:, -1, :])$$

   w_t is the word embedding of the t^{th} token
   h_t is the hidden state at t^{th} token.
   output is the output features (h_t) from the last layer. Size([batch size, seq len, hidden size])
   cls is the classification output Size([batch size, seq len, cls dim]

2) performance of your model.

   Score: 0.90177

3) the loss function you used.

   我採用 cross entropy 做爲我的 loss function.

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

   我採用 Adam 做爲我的 optimization algorithm。learning rate 我採用 0.001，並採用 PyTorch 的 ReduceLROnPlateau 函式在 evaluation loss 收斂時縮小 learning rate，讓模型的 loss 盡可能地變小。batch size 我設定爲 128。

Question 3. Describe your slot tagging model.

1) Describe your model

我將每個 batch 的資料先過一層 dropout layer(p=0.5)，之後接一個 LSTM(hidden size=512, num layers=3, dropout=0.5, bidirectional=True)，再接一個 linear layer 去預測分類結果。

$$w_t = DROPOUT(w_t)$$
$$output, (h_t, c_t) = LSTM(w_t, h_{t-1}, c_{t-1})$$
$$cls = LINEAR(output)$$
$$cls1 = cls.permute(0, 2, 1)$$

w_t is the word embedding of the t^{th} token
h_t is the hidden state at t^{th} token.
output is the output features (h_t) from the last layer. Size([batch size, seq len, hidden size])
cls is the classification output. Size([batch size, seq len, cls dim]
$cls1$ is the reshape of cls. Size([batch size, cls dim, seq len]

2) performance of your model.
Score: 0.78069

3) the loss function you used.
我採用 cross entropy 做爲我的 loss function.

4) The optimization algorithm (e.g. Adam), learning rate and batch size.
我採用 Adam 做爲我的 optimization algorithm。learning rate 我採用 0.001，並採用 PyTorch 的 ReduceLROnPlateau 函式在 evaluation loss 收斂時縮小 learning rate，讓模型的 loss 盡可能地變小。batch size 我設定爲 128。

Question 4. Sequence Tagging Evaluation

1) Token accuracy: 0.97592
2) Joint accuracy: 0.85700

	precision	recall	f1-score	support
date	0.76	0.77	0.77	206
first_name	0.95	0.94	0.95	102
last_name	0.88	0.59	0.71	78
people	0.74	0.75	0.74	238
time	0.83	0.85	0.84	218
micro avg	0.80	0.79	0.80	842
macro avg	0.83	0.78	0.80	842
weighted avg	0.81	0.79	0.80	842

FIGURE 1. IOB2 report.

Recall(召回率) = TP/(TP+FN)
Precision(準確率) = TP/(TP+FP)
F1-score = 2 * Precision * Recall / (Precision + Recall)

support 爲各分類樣本的總數量
我們可以看到"last name" 的樣本數較少，因此會有 data imbalance 的問題，對於這個模型來說 Precision 高而 Recall 較低，可以看作一個比較謹慎的模型，雖然常常沒辦法抓出"last name" ，但只要有抓出幾乎都是正確的，而 F1-score 可以看做兩者的調和平均數。

Question 5. Compare with different configurations

在 intent 分類任務上，我嘗試過用、加 dropout 與否、不同 hidden size 及不同 layer 的 LSTM, GRU 模型。後來發現在 embedding 後先加入一層 dropout 可以有效改善 overfitting 的問題，而 GRU 在這個分類任務上的表現也比 LSTM 好。其中一個特別注意的事我原先在我的模型會加入一層 softmax，後來發現拿掉後收斂的速度快了很多，後面研究了 pytorch 才知道原來乎叫 crossentropy 函式時會自已做 softmax，也就是說如果加入 softmax 等於做了 2 次 softmax，這樣會使得 output 的值變的比較平均，因此使得 training 較不容易收斂，甚至在某些參數的設定下 train 了 2000 個 epoch 都不會收斂。

在 slot tagging 分類任務上，有了 intent 分類的經驗，我特別注意了不要添加 softmax 層，並且在 embedding 後先加入一層 dropout 在放入 rnn，在 slot tagging 的任務上，我發現 LSTM 比 GRU 的表現還要好，因此我選用 LSTM 做爲我的模型。另外在直覺上如果要對一個單字進行詞性分類考慮前後文應該會得到比較好的 performance，因此我採用 bidirectional 的 LSTM。

R10521601