LSTM and GRU models for Intent Classification & Slot Tagging

目錄

Q1: Data processing

我利用 simple code 的方法做資料前處理：

Intent 及 slot tagging 的訓練資料透過 preprocess_intent.py 及 preprocess_slot.py 利用 GloVe(Global Vectors for Word Representation)生成 Intent 及 slot tagging 的 embeddings.pt。GloVe 模型結合語言資料庫全域性統計（count-based & overall statistics）的特徵和 word2vec 文字局部的上下文的特徵。embeddings.pt 的目的是要將訓練資料的每個文字(token)用 300 維的向量表示其特徵(feature)。

將訓練及驗證資料放進 embeddings.pt 前需要將文字轉成編碼(token2id)讓模型認得文字，我使用 utils.py 中的 encode 函式完成。文字轉成編碼後，我將資料放進 DataLoder，但每筆訓練資料的文字長度不同，所以需要用 DataLoder 中 collate_fn 的功能將每個 batch 中的資料調整成相同長度才能放進模型做訓練。進入自行設計的模型前，每個 batch 會先進入 embeddings.pt 中將文字抓取特徵，公式(1)所示

x = Embedding(batch) 公式(1)

說明：batch 維度: [batch size, sequence length]；x 維度: [batch size, sequence length, 300(feature)]。

資料處理流程如圖 1 所示：

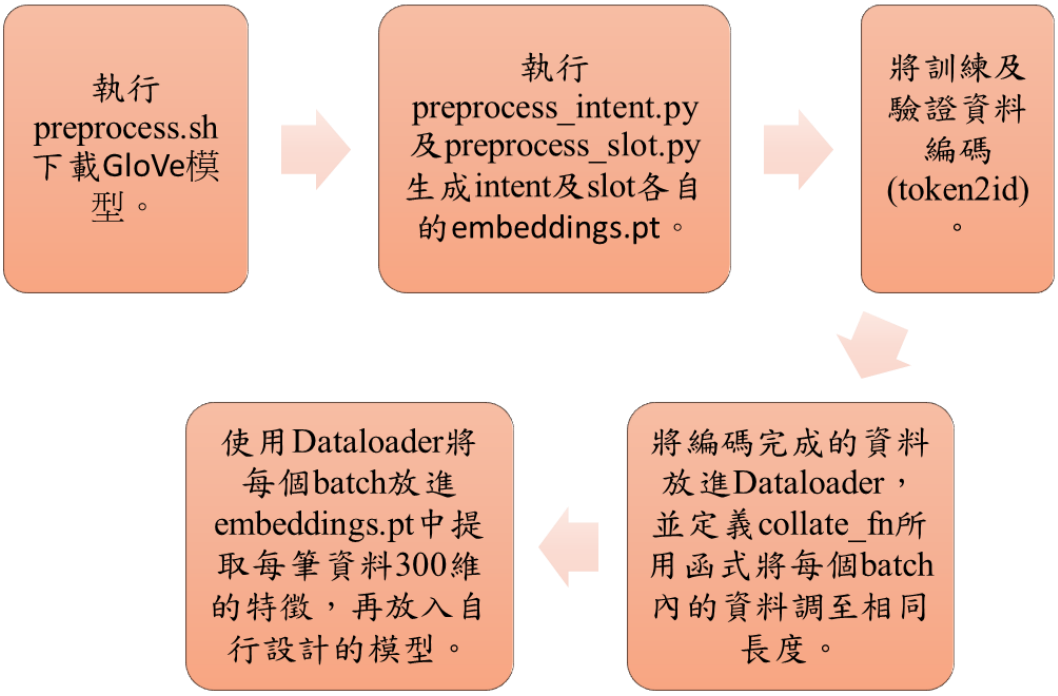

圖 1、Data processing 流程

Q2: Describe your intent classification model

a. My model

模型架構：

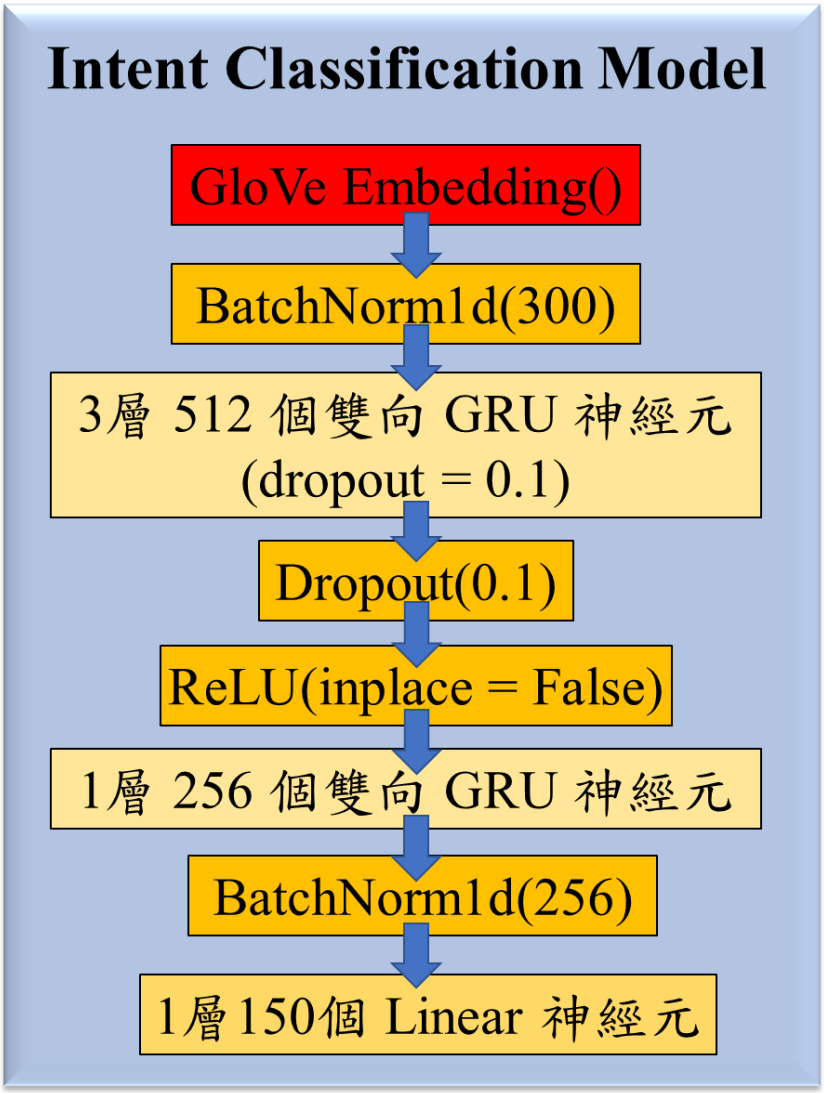

圖 2、intent classification 模型架構

x_0 = Embedding(batch) 說明: batch 變數為 Dataloader 中資料的每個 batch；Embedding()將文字資料提取特徵。

x_1 = x_0.permute(0, 2, 1) 說明: permute ()將 x_0 維度轉換成符合 BatchNorm1d 的格式

x_2 = BatchNorm1d_1(x_1) 說明: BatchNorm1d_1()將 x_1 做正規化。

x_3 = x_2.permute(0, 2, 1) 說明: permute ()將 x_2 維度轉換成符合 BatchNorm1d 的

格式。

x_4, $h1_t$ = GRU1(x_3, $h1_{t-1}$) 說明: GRU1()為 GRU 神經網絡。

x_5 = Dropout(x_4) 說明: Dropout()為 Dropout 層 。

x_6 = ReLU(x_5) 說明: ReLU ()為 Rectified Linear Unit。

x_7, $h2_t$ = GRU2(x_6, $h2_{t-1}$) 說明: GRU1()為 GRU 神經網絡。

out_0 = BatchNorm1d_2($h2_t$[-1]) 說明: BatchNorm1d_2()將 $h2_t$[-1]做正規化。

out_1 = Linear(out_0) 說明: Linear ()為 Linear 層 。

參數設置：

BatchNorm1d_1 = torch.nn.BatchNorm1d(300)

GRU1 = torch.nn.GRU(

input_size =300 ,

hidden_size = 512,

batch_first = True,

num_layers =3,

dropout = 0.1,

bidirectional = True)

Dropout = torch.nn.Dropout(0.1)

ReLU = torch.nn.ReLU(inplace = False)

GRU2 = torch.nn.GRU(

input_size = 1024,

hidden_size = 256,

batch_first = True,

num_layers = 1,

dropout = 0,

bidirectional = True)

BatchNorm1d_2 = torch.nn.BatchNorm1d(256)

Linear = torch.nn.Linear(in_features = 256, out_features = 150)

b. Performance of my model

Public: 0.90444

c. The loss function I used

我使用 CrossEntropyLoss()函數(如公式(2))做為 loss function，可用來量化兩個概率分佈之間差異的損失函數，多用於分類問題，pytorch 的 CrossEntropyLoss 結合了 LogSoftmax 和 NLLLoss。

$$l_n = -\sum_{c=1}^{C} w_c \log \frac{\exp(x_{n,c})}{\sum_{i=1}^{C} \exp(x_{n,i})} y_{n,c} \qquad \text{公式(2)}$$

說明：x 為預測值；y 為實際值；w 為權重；C 為分類數量；n 為第幾個 batch

d. The optimization algorithm, learning rate and batch size

- ✓ Optimization algorithm 我採用 Adam。
- ✓ Learning rate 我設為 10^{-3}。
- ✓ Batch size 我設為 128。

Q3: Describe your slot tagging model

a. My model

模型架構：

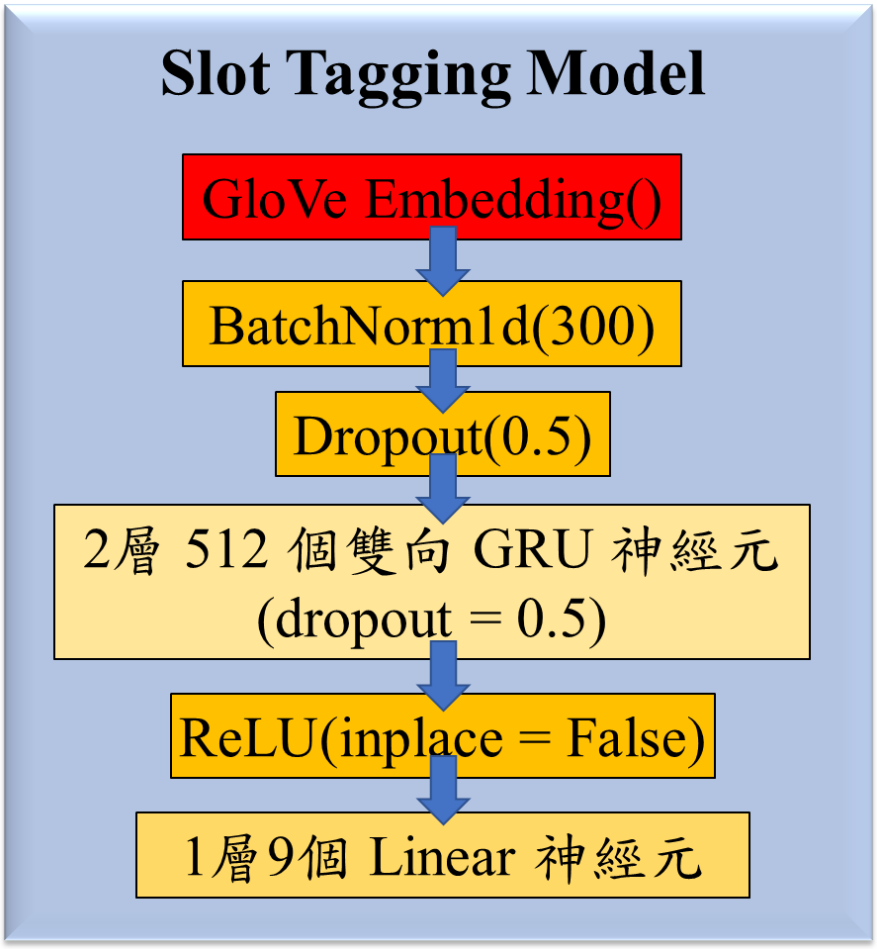

圖 3、slot tagging 模型架構

x_0 = Embedding(batch) 說明: batch 變數為 Dataloader 中資料的每個 batch；Embedding()將文字資料提取特徵。

x_1 = x_0.permute(0, 2, 1) 說明: permute ()將 x_0 維度轉換成符合 BatchNorm1d 的格式

x_2 = BatchNorm1d_1(x_1) 說明: BatchNorm1d_1()將 x_1 做正規化。

x_3 = x_2.permute(0, 2, 1) 說明: permute ()將 x_2 維度轉換成符合 BatchNorm1d 的格式。

x_4 = Dropout(x_3) 說明: Dropout()為 Dropout 層 。

x_5, $h1_t = GRU1(x_4, h1_{t-1})$ 說明: GRU1()為 GRU 神經網絡。

$x_6 = ReLU(x_5)$ 說明: ReLU ()為 Rectified Linear Unit。

Out = Linear(x_6) 說明: Linear ()為 Linear 層。

參數設置：

BatchNorm1d_1 = torch.nn.BatchNorm1d(300)

Dropout = torch.nn.Dropout(0.5)

GRU1 = torch.nn.GRU(

input_size =300 ,

hidden_size = 512,

batch_first = True,

num_layers =2,

dropout = 0.5,

bidirectional = True)

ReLU = torch.nn.ReLU(inplace = False)

Linear = torch.nn.Linear(in_features = 1024, out_features = 9)

b. Performance of my model

Public: 0.83914

c. The loss function I used

我使用 CrossEntropyLoss()函數(如公式(2))做為 loss function，可用來量化兩個概率分佈之間差異的損失函數，多用於分類問題，pytorch 的 CrossEntropyLoss 結合了 LogSoftmax 和 NLLLoss。

d. The optimization algorithm, learning rate and batch size

- ✓ Optimization algorithm 我採用 Adam。
- ✓ Learning rate 我設為 10^{-3}。
- ✓ Batch size 我設為 128。

Q4: Sequence Tagging Evaluation

Seqeval 計算 precision, recall, f1-score 只會將文字類別為 B 開頭的做計算，本次的訓練資料有："B-last_name", "B-date", "B-time", "B-first_name", "B-people" 這五種，但計算 token accuracy 和 joint accuracy 時，會利用所有類別的資料做計算。所有計算公式如下：

$$\text{Token Accuracy} = \frac{\text{預測文字類別正確的個數}}{\text{文字的個數}} \left(\frac{\text{TP+TN}}{\text{TP+FP+FN+TN}}\right) \quad \text{公式(3)}$$

$$\text{Joint Accuracy} = \frac{\text{預測文字類別都正確的句子個數}}{\text{句子的個數}} \quad \text{公式(4)}$$

$$\text{Precision} = \frac{\text{預測 B 類別文字正確的個數}}{\text{預測 B 類別文字的個數}} \left(\frac{\text{TP}}{\text{TP+FN}}\right) \quad \text{公式(5)}$$

$$\text{Recall} = \frac{\text{預測 B 類別文字正確的個數}}{\text{標註為 B 類別文字的個數}} \left(\frac{\text{TP}}{\text{TP+FN}}\right) \quad \text{公式(6)}$$

$$\text{F1} - \text{score} = \frac{2 * \text{Token Accuracy} * \text{Recall}}{\text{Token Accuracy} * \text{Recall}} \quad \text{公式(7)}$$

說明：

True Positive: TP，真實情況為 True,預測也為正的樣本數。

False Positive:FP，真實情況為 False，預測為正的樣本數。

False Negative:FN，真實情況為 True，預測為負的樣本數。

True Negative: TN，真實情況為 False，預測為負的樣本數。

以 **Q3** 介紹的模型展示以上評估指標在每個 epoch 訓練後以驗證資料(eval.json)計算的結果：

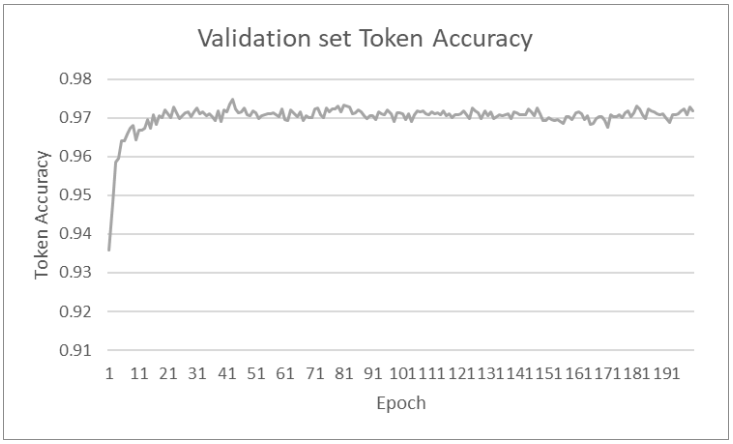

圖 4、Token Accuracy

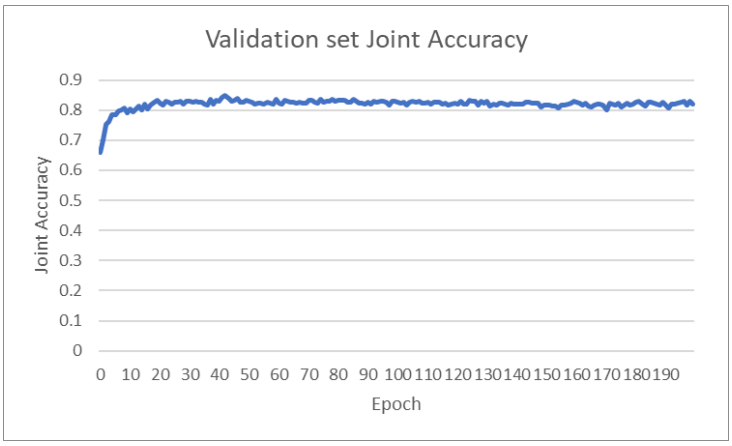

圖 5、Joint Accuracy

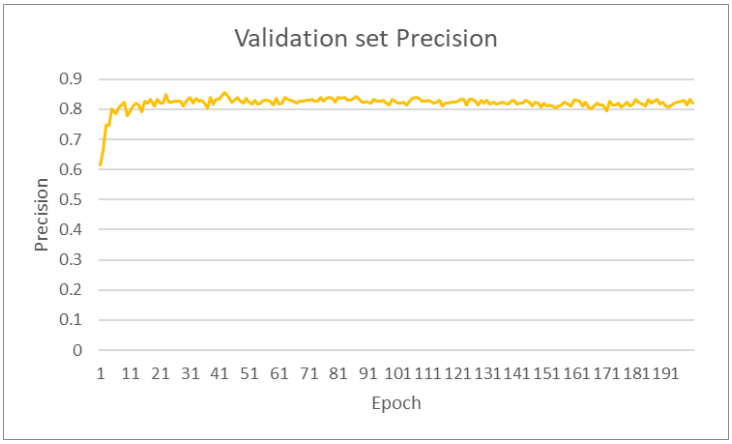

圖 6、Precision

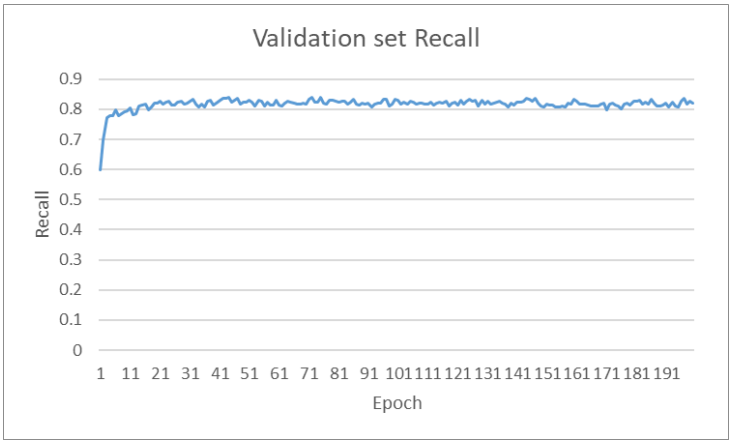

圖 7、Recall

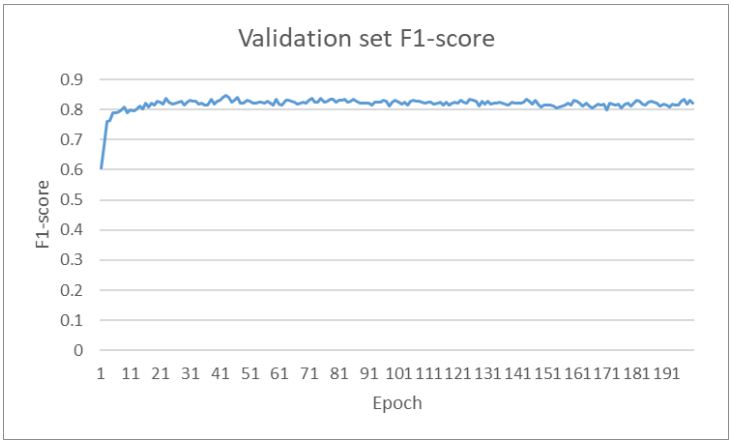

圖 8、F1 score

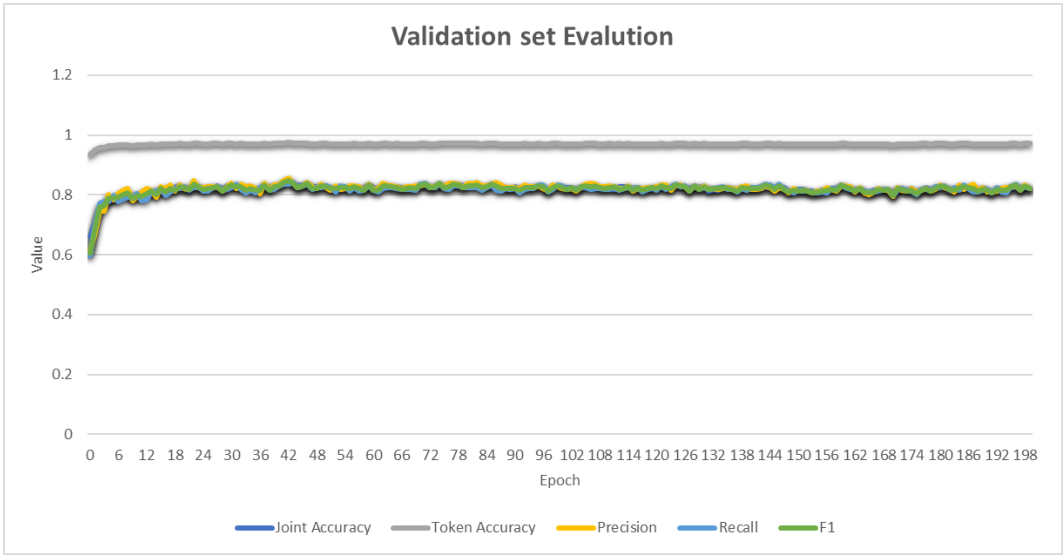

圖 9、All Evalution function result

此模型訓練時在第 43 個 epoch 時出現 Joint Accuracy 最高的值 0.849。由圖 9 可以發現除了 Token Accuracy 之外，其他評估指標的趨勢及值相似。

表 1、Seqeval Classification Report

Epoch 43	precision	recall	f1-score	support
date	0.796954	0.762136	0.779156	206
first_name	0.960396	0.95098	0.955665	102
last_name	0.929577	0.846154	0.885906	78
people	0.805907	0.802521	0.804211	238
time	0.894009	0.889908	0.891954	218
micro avg	0.856622	0.837292	0.846847	842
macro avg	0.877369	0.85034	0.863378	842
weighted avg	0.856698	0.837292	0.846714	842

表 1 為 Seqeval Classification Report 在第 43 個 epoch 時的情形，在三個指

標的 macro avg 和 micro avg 差異在 0.02 左右，不會差異過大，但其中 date 類別資料在三個評估指標中皆是最低的，若想要優化模型，可以從 date 類別的資料著手。

Q5: Compare with different configurations

a. LSTM vs. GRU

一開始選擇模型時有固定 LSTM 及 GRU 的參數設定，結果 GRU 在 intent 及 slot tagging 的任務上皆表現比 LSTM 較好，代表本次資料集不適合參數量較多的 LSTM。

b. Linear 層數設定

固定使用 GRU 後，針對 Linear 層數有嘗試接一層或兩層再輸出，實測接一層 Linear 後直接輸出的預測結果較好。

c. BatchNorm1d 使用

BatchNorm1d 可將輸入的特徵正規化且在實際訓練上可以加快模型收斂時間，及提高模型預測準確性，intent 及 slot tagging 的任務上我皆有在 Embedding 結束後加上 BatchNorm1d。

d. 利用 GRU 的 Output 最後一層或 h_t 最後一層的結果往 Linear 層傳

Intent 任務中，可以選擇用 GRU 的 Output 最後一層或 h_t 最後一層做為 GRU 的輸出，這兩個輸出的 feature 數量會差一倍，因為有開雙向的關係。實測中，用 h_t 最後一層輸出對模型預測的表現較好。

e. GRU 模型架構調整

我分別調整 Intent 及 Slot tagging 的 GRU layer 數量 1 至 4 個：在 intent 中，layer 設定為 4 表現最好；在 slot tagging 中，layer 設定為 2 表現較好。GRU hidden size 我嘗試分別調 128、256 及 512，最後 512 效果最好。

我將 GRU 拆成兩個不同的 hidden size，一個 hidden size 設為為 512，另一個 hidden size 設為為 256，Embedding 結束後先通過 hidden size 為 512 的 GRU 再通過 hidden size 為 256 的 GRU。在 intent 中拆成兩個 GRU 的表現較好；但 slot tagging 這樣拆的表現較差。

f. Dropout 設定

我在 intent 及 slot tagging 中分別對 dropout 值的設定在 0 至 0.7 之間每 0.1 測一次。Dropout 的設定和模型的複雜度有關聯：我最後做出的 Intent 的模型層數較少，dropout 設定在 0.1 或 0.2 對模型訓練效果較好；slot tagging 的模型層數較多，dropout 設定在 0.4、0.5 或 0.6 對模型訓練效果較好。

g. Batch size 及 Learning rate 設定

Batch size 我調過 64、128、256，沒有明顯看出差異；Learning rate 我調過 10^{-2}、10^{-3}、10^{-4} 及 10^{-5}，使用 10^{-2} 會讓模型無法降低 loss，推測為模型參數一次調整的量過大，10^{-3}、10^{-4} 及 10^{-5} 皆可使模型的 loss 降低，10^{-3} 訓練效果最好。