Q1:(use sample code)

A:collect all words in train and valid set,and get the representation(size=300) from glove model,if the word in glove,return the corresponding representation(size=300), else return the random representation([random() * 2 - 1 for _ in range(glove_dim)])

B:form a table with the representations collected in A

Q2:

a. My model architecture:An Embedding layer with pretrained weight,than a GRU module with input size(300),hidden_size(128),two layer,dropout(0.4),bidirectional(true),and a mlp module composed of two linear layer(hiddensize*2->hiddensize*2,hidden_size*2->num_class) with activate function LeakyReLU to project the output of GRU into num_class(intent_classify=150,slot_tagging=10)

   My forward function:feed tokenized text into Embedding with output size(batch_size,max_len,300),and than feed into GRU module to calculate h_t, c_t = GRU(w_t, h_{t-1}, c_{t-1}),where w_t is the t-th token in the text(size=300),t=1~128(maxlen), h_{t-1} is (t-1)th hidden state,ct is the output of (i-1)th token ,than get mean value $(\sum_1^{128} wt)$/128,and feed the output into mlp module and output the probability distribution with size(num_class=150)

b. 0.90311

c. Crossentropy(input(size=150),label(one_hot_vector))

d. use Adam(lr=2e-5), batch_size=2

Q3:

c. My model architecture:An Embedding layer with pretrained weight,than a GRU module with input size(300),hidden_size(128),two layer,dropout(0.4),bidirectional(true),and a mlp module composed of two linear layer(hiddensize*2->hiddensize*2,hidden_size*2->num_class) with activate function LeakyReLU to project the output of GRU into num_class(intent_classify=150,slot_tagging=10)

   My forward function:feed tokenized text into Embedding with output size(batch_size,max_len,300),and than feed into GRU module to calculate h_t, c_t = GRU(w_t, h_{t-1}, c_{t-1}),where w_t is the t-th token in the text(size=300),t=1~128(maxlen), h_{t-1} is (t-1)th hidden state,ct is the output of (i-1)th token,than feed the output into mlp module and output the probability distribution with size(num_class=150)

a. 0.77533

Crossentropy(input(size=10),label(one_hot_vector))

b. use Adam(lr=2e-5), batch_size=2

Q4:

	precision	recall	f1-score	support
date	0.72	0.73	0.73	203
first_name	0.81	0.93	0.87	89
last_name	0.68	0.70	0.69	76
people	0.66	0.72	0.69	217
time	0.85	0.87	0.86	214
micro avg	0.75	0.79	0.77	799
macro avg	0.75	0.79	0.77	799
weighted avg	0.75	0.79	0.77	799

Token acc : (count of right tokens)/(number of all tokens)

Joint acc : (count of right texts)/(number of all texts)

Seqval:

Calculate Precision and Recall for each type of tokens in the data

Precision:(number of samples with token i)/(number of samples predicted as token i)

Recall: [number of samples(actually is token i) predicted as token i]/(number of samples with token i)

F1-score:2*(Precision*Recall)/(Precision+Recall)

Q5:測試 GRU LSTM RNN

Intent

Epoch:40

Lr:2e-5

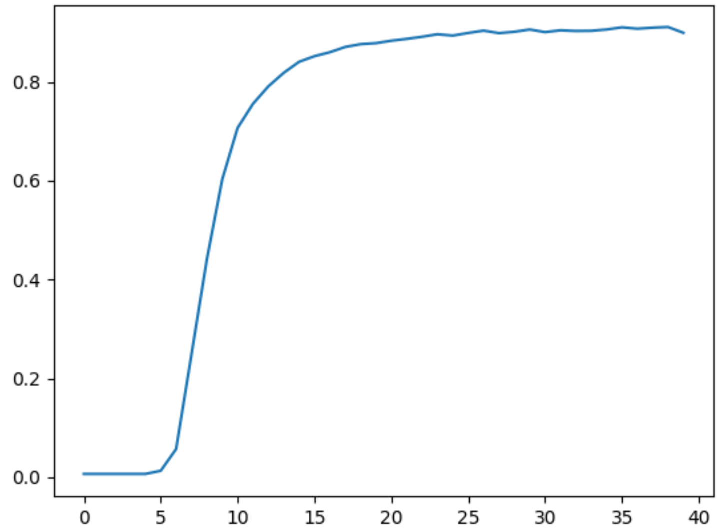
Optimizer:Adam

Batchsize:2

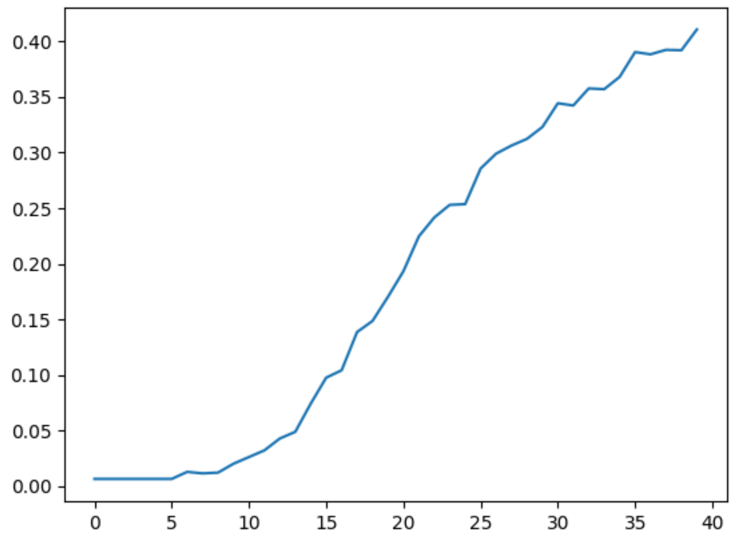
Layer:2

Bidirectional:true

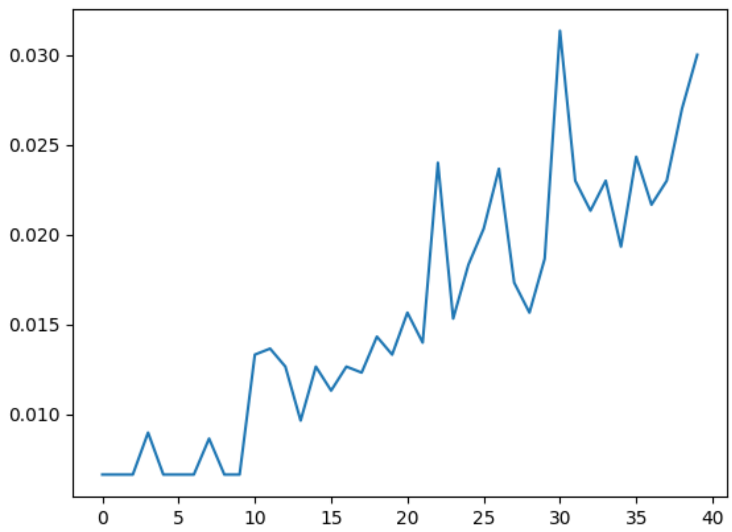
GRU:



LSTM:

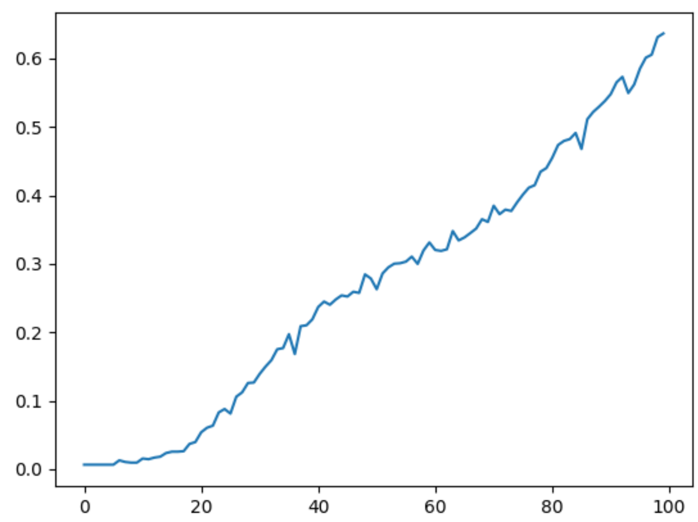


RNN:

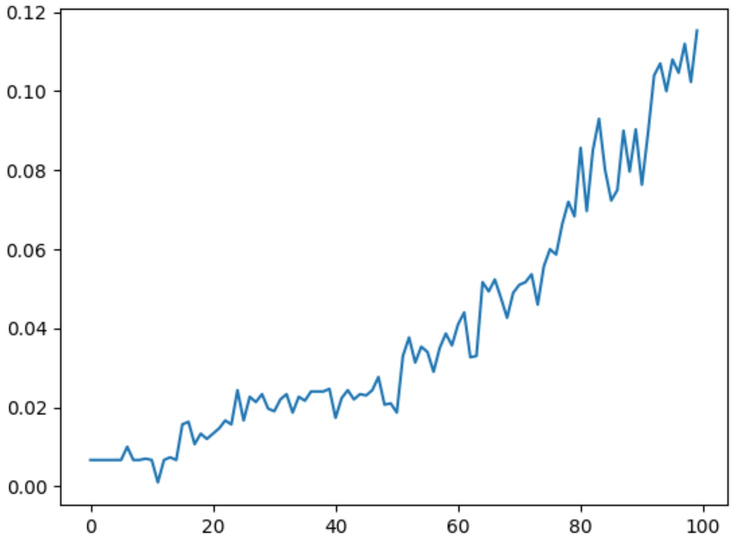


因 LSTM 和 RNN 還沒收斂，因此將 epoch 調至 100，其餘參數不變

LSTM:

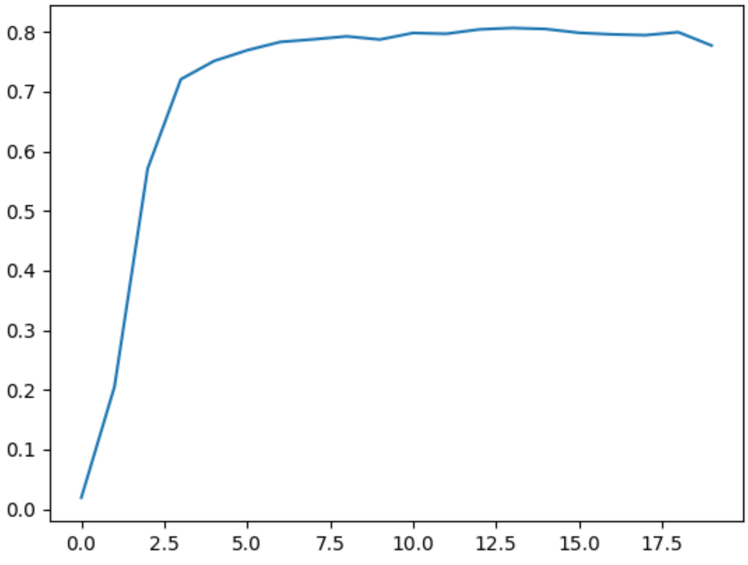


RNN:

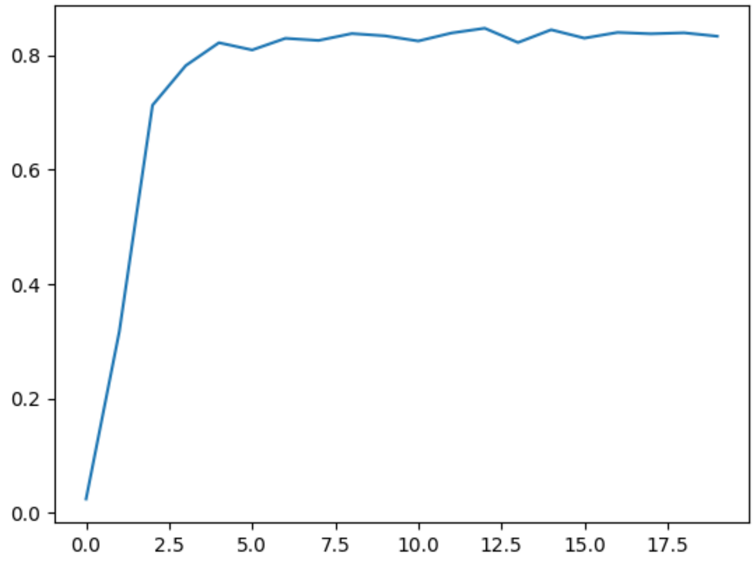


由以上的比較可知，在 epoch 以外其餘條件相同的情況下，LSTM,RNN 收斂速度都較 GRU 慢，且 100 個 epoch 仍不足以讓他們收斂，因此我將 lr 改成了 2e-3，重跑了三個 model。(因時間關係都只跑了 20 epoch)

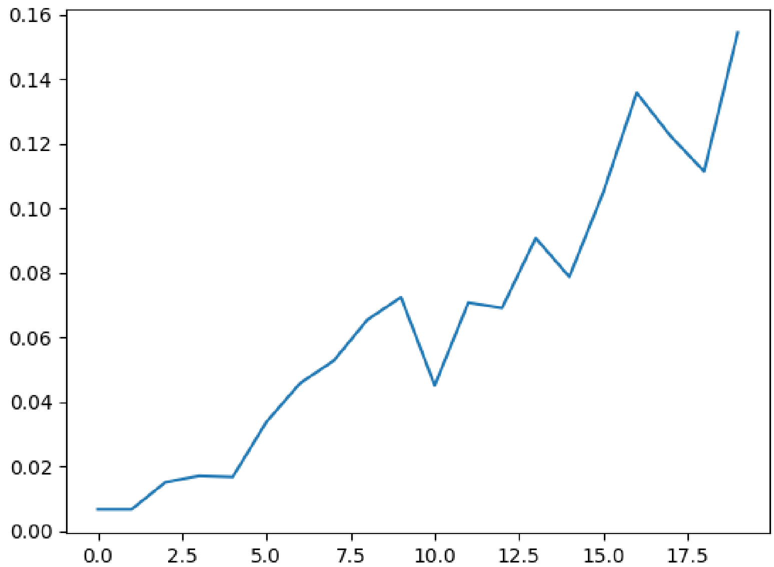
GRU:



LSTM:


0.0
0.2
0.4
0.6
0.8
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

RNN:


0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

Slot

Epoch:40

Lr:2e-5

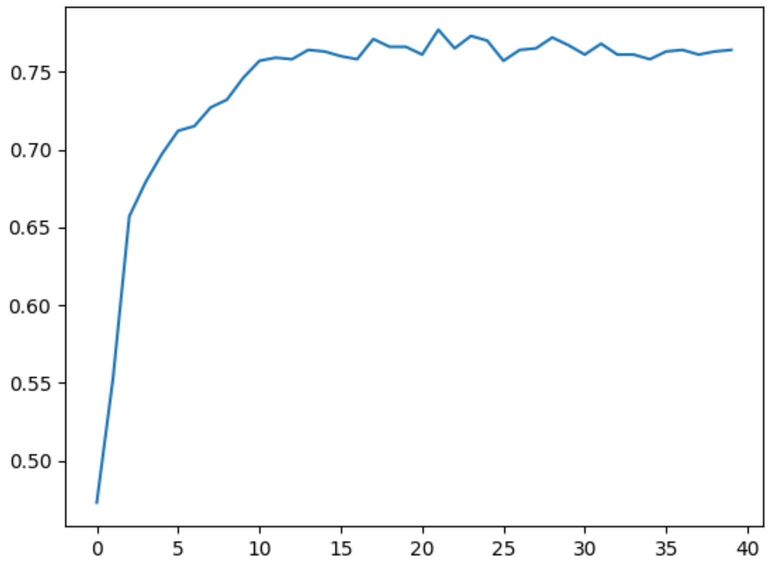
Optimizer:Adam

Batchsize:2

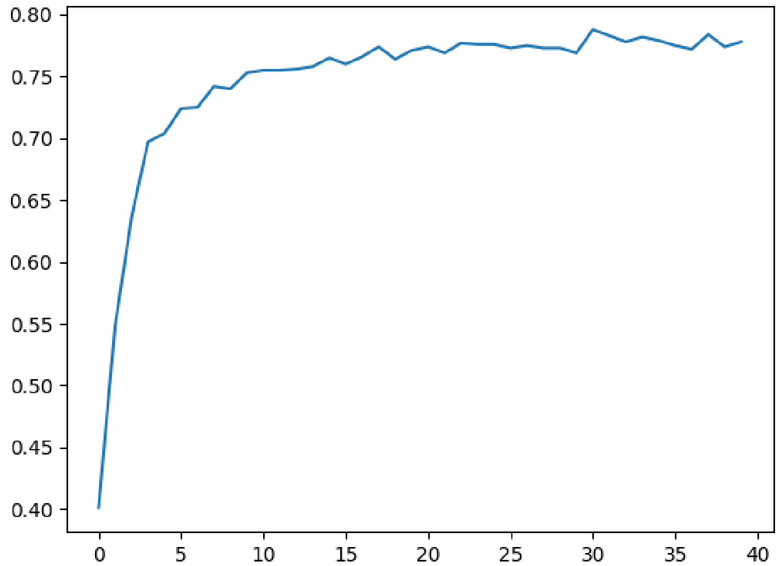
Layer:2

Bidirectional:true

GRU:



LSTM:



由以上可知在此 slot_tagging 中，GRU 收斂的反而比 LSTM 漫一點，而 LSTM 在 valid_set 的表現則稍佳(因 RNN 效果不佳故無放上圖表)