# Q1: Data processing (2%)

1. Describe how do you use the data for `intent_cls.sh, slot_tag.sh`:
    a. How do you tokenize the data?
    使用sample code進行資料處理，用空白字來去區分文字，再計算每個詞在dataset中出現幾次。
    b. The pre-trained embedding you used.
    sample code的Glove embedding
2. If you use the sample code, you will need to explain what it does in your own ways to answer Q1.
    利用 glove.840B.300d.txt

# Q2: Describe your intent classification model. (2%)

1. Describe
    a. your model
    利用sample code的glove embedding將資料轉成詞向量，接著將embedding傳入LSTM

    $$out, (h_t, c_t) = LSTM(input, (h_0, c_0))$$

    接著進入batch_norm和layer_norm，將結果輸入3層linear layer，dropout為預設值0.1，activation function為relu，最後轉成維度為num_class的輸出。
    b. performance of your model.
    Public score: 0.75710
    Private score: 0.76420
    c. the loss function you used.
    nn.CrossEntropyLoss()
    d. The optimization algorithm (e.g. Adam), learning rate and batch size.
    Optimizor: Adam
    Learning rate: 0.0001
    Batch size: 128
    Training epoch: 100
    Hidden_size: 512
    Num_layers :2

# Q3: Describe your slot tagging model. (2%)

1. Describe
    a. your model
    利用example code的glove embedding將資料轉成詞向量，接著將embedding傳入layer_norm，接著將正規化後的輸出傳入GRU

    $$out, (h_t, c_t) = GRU(input, (h_0, c_0))$$

接著進入batch_nrom和layer_norm，將結果輸入1層linear layer，dropout為預設值0.1，activation function為relu，最後轉成維度為num_class的輸出。

b. performance of your model.

Public score: 0.75710

Private score: 0.76420

c. the loss function you used.

nn.CrossEntropyLoss()

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

Optimizor: Adam

Learning rate: 0.0001

Batch size: 128

Training epoch:25

Hidden_size: 512

Num_layers :2

## Q4: Sequence Tagging Evaluation (2%)

- Please use seqeval to evaluate your model in Q3 on validation set and report *classification_report(scheme=IOB2, mode='strict')*.
- Explain the differences between the evaluation method in seqeval, token accuracy, and joint accuracy.

seqeval 評估結果

準確率(accuracy) = 正確預測的元素個數 / 總元素個數

精準率(precision) = 正確預測的實體個數 / 預測的總實體個數

召回率(recall) = 正確預測的實體個數 / label 的總實體個數

F1值(F1 scroe) = 2 * 準確率 * 召回率 / (準確率 + 召回率)

classification report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| date | 0.73 | 0.70 | 0.72 | 206 |
| first_name | 0.94 | 0.89 | 0.91 | 102 |
| last_name | 0.79 | 0.69 | 0.74 | 78 |
| people | 0.69 | 0.68 | 0.68 | 238 |
| time | 0.84 | 0.87 | 0.85 | 218 |
| micro avg | 0.78 | 0.76 | 0.77 | 842 |
| macro avg | 0.80 | 0.77 | 0.78 | 842 |
| weighted avg | 0.78 | 0.76 | 0.77 | 842 |

token_accuracy為單一token預測tag正確即計算一次

joint_accuracy 為整句話的tag均預測成功才算成功

訓練初期的token_accuracy較高但joint_accuracy較低，整體預測準確率還太低。

```
Epoch: 2
100%|██████████| 57/57 [00:21<00:00,  2.68it/s]
loss:  0.05322668328881264   joint_acc:  0.4014356709000552    token_acc:  0.9887536236885699
100%|██████████| 8/8 [00:01<00:00,  6.65it/s]
loss_eval:  0.05060149356722832     joint_acc_eval:  0.363  token_acc_eval:  0.7086899503036996
trigger: 0
Epoch: 3
100%|██████████| 57/57 [00:21<00:00,  2.68it/s]
loss:  0.04840999096632004   joint_acc:  0.4014356709000552    token_acc:  0.9887536236885699
100%|██████████| 8/8 [00:01<00:00,  6.87it/s]
loss_eval:  0.04539646580815315     joint_acc_eval:  0.361  token_acc_eval:  0.7085120013114301
trigger: 0
Epoch: 4
100%|██████████| 57/57 [00:21<00:00,  2.67it/s]
loss:  0.0417783223092556   joint_acc:  0.4014356709000552    token_acc:  0.988955299213142
100%|██████████| 8/8 [00:01<00:00,  6.86it/s]
loss_eval:  0.03730613365769386     joint_acc_eval:  0.36  token_acc_eval:  0.7094470423799006
trigger: 0
Epoch: 5
100%|██████████| 57/57 [00:21<00:00,  2.61it/s]
loss:  0.033587951213121414   joint_acc:  0.4131695196024296    token_acc:  0.9904090885560464
100%|██████████| 8/8 [00:01<00:00,  6.62it/s]
loss_eval:  0.029548054561018944     joint_acc_eval:  0.387  token_acc_eval:  0.7105179717697405
trigger: 0
Epoch: 6
100%|██████████| 57/57 [00:21<00:00,  2.63it/s]
loss:  0.026278100907802582   joint_acc:  0.45278851463279957    token_acc:  0.9923406439812258
100%|██████████| 8/8 [00:01<00:00,  6.83it/s]
loss_eval:  0.023158863186836243     joint_acc_eval:  0.427  token_acc_eval:  0.7118401867062396
```

# Q5: Compare with different configurations (1% + Bonus 1%)

- Please try to improve your baseline method (in Q2 or Q3) with different configurations (includes but not limited to different number of layers, hidden dimension, GRU/LSTM/RNN) and EXPLAIN how does this affects your performance / speed of convergence / ...
- Some possible BONUS tricks that you can try: multi-tasking, few-shot learning, zero-shot learning, CRF, CNN-BiLSTM
- This question will be graded by the completeness of your experiments and your findings.