ADL Homework 1 Report

Name: Chou Tun Hsiang **Student Number:** B07902050 **Date:** April 13 2021

1 Data Preprocessing

The sample code given is used for data preprocessing. There are mainly three things we need to do before training a model. The first thing is to map each 'intent' in the intent classification task or each 'tag' in the slot tagging task into an unique identifier. The second thing is to collect every word in training and developing datasets and build a dictionary which helps us to convert each word to an index. The last thing is to load the pretrained embeddings and build an embedding matrix according to the word-index relationship we just created. With the embedding matrix, we can represent each word with a vector based on their semantic similarity. The more similar the meaning of two words, the closer their vectors would be in the embedding space. One thing to note is that not every word in the dataset owns a corresponding embedding vector in the pretrained embeddings. In this occasion, we simply generate a random vector to represent the word.

2 Intent Classification Model

- public score on kaggle: 0.90088
- loss function: cross entropy
- optimization algorithm: Adam
- model structure

```
1. Word embedding with 300 dimensions
2. LSTM(hidden_size=512, num_layers=2, bidirectional=True, dropout=0)
3. ReLU()
4. Linear(in_features=1024, out_features=64, bias=True)
5. ReLU()
6. Linear(in_features=64, out_features=150, bias=True)
```

3 Slot Tagging Model

- public score on kaggle: 0.81608
- loss function: cross entropy

- optimization algorithm: Adam
- model structure

```
1. Word embedding with 300 dimensions
2. LSTM(hidden_size=512, num_layers=2, bidirectional=True, dropout=0)
3. ReLU()
4. Linear(in_features=1024, out_features=64, bias=True)
5. ReLU()
6. Linear(in_features=64, out_features=9, bias=True)
```

4 Sequence Tagging Evaluation

```
                precision    recall  f1-score   support

          date       0.81      0.80      0.80       206
    first_name       0.91      0.94      0.93       102
     last_name       0.92      0.85      0.88        78
        people       0.76      0.75      0.76       238
          time       0.84      0.78      0.81       218

     micro avg       0.83      0.80      0.81       842
     macro avg       0.85      0.82      0.84       842
  weighted avg       0.83      0.80      0.81       842

joint accuracy: 0.817
token accuracy: 0.969332150551261
```

- precision $= \dfrac{\text{TP}}{\text{TP} + \text{FP}}$
- recall $= \dfrac{\text{TP}}{\text{TP} + \text{FN}}$
- f1 score $= \dfrac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$
- support: number of occurences of each class
- micro average: calculate metrics globally by counting the total true positives, false negatives and false positives
- macro average: calculate metrics for each label, and find their unweighted mean

With metrics like precision, recall, and f1-score, we can see how the model perform on each class, as well as how many positive instances are mispredicted as negative, and vice versa, while joint accuracy and token accuracy give us an overall impression of how much the model has learned on this task.

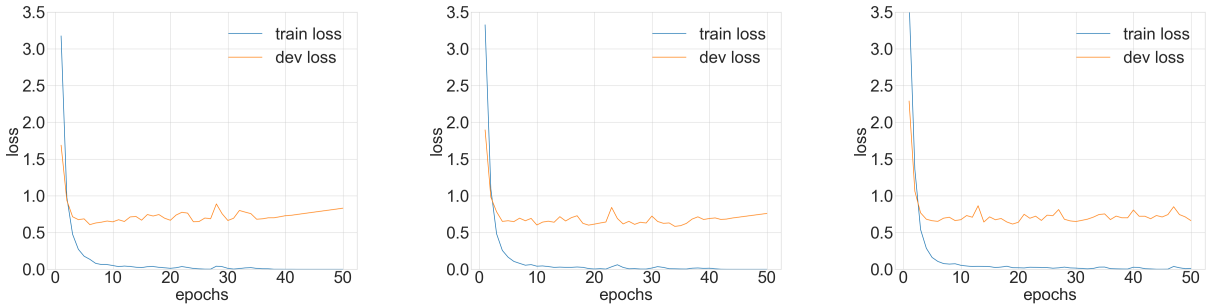
5 Compare with Different Configurations

5.1 Intent Classification

The line plots below illustrate the training loss and evaluating loss of LSTM model with different configurations. The three models are all bidirectional, and trained with Adam and cross entropy loss. The model begins to converge after about 10 epochs. It seems that the number of hidden size or layers does not have a great impact over the learning process.

	(a)	(b)	(c)
hidden size	256	512	512
layer	1	1	2
learning rate	1e-3	1e-3	1e-3
bidirectional	True	True	True

Table 1: Model configurations



(a) (b) (c)

Figure 1: LSTM with Adam and cross entropy loss

5.2 Slot Tagging

Below are figures showing training and evaluating loss curve of LSTM model with configurations specified in the table below. All models are trained with Adam and cross entropy loss. In general, while the training loss declines smoothly, the validation loss tends to fluctuate as

the training proceeds. This may imply that LSTM models easily overfit on the task. From the figures we can see that bidirectional LSTMs have a better generalization performance, if we compare (a) with (d), (b) with (e), and (c) with (f) together, whereas increasing the number of hidden size (figure a. and figure c.) or doubling the number of layers (figure a. and figure b.) could cause the learning process to deteriorate with unidirectional LSTM. However, with bidirectional LSTM, two-layer model seems to outperform single layer's (figure d. and figure e.). Decreasing the learning rate also stabilizes the variation of the evaluation loss (figure a. and figure g.) and prevent the model from overfitting. Other tactics such as dropout or regularization can be taken into account as well, but more experiments need to be done to verify the effectiveness.

	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)
hidden size	512	512	1024	512	512	1024	512	512
layer	1	2	1	1	2	1	1	1
learning rate	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3	1e-4	1e-2
bidirectional	False	False	False	True	True	True	True	True

Table 2: Model configurations

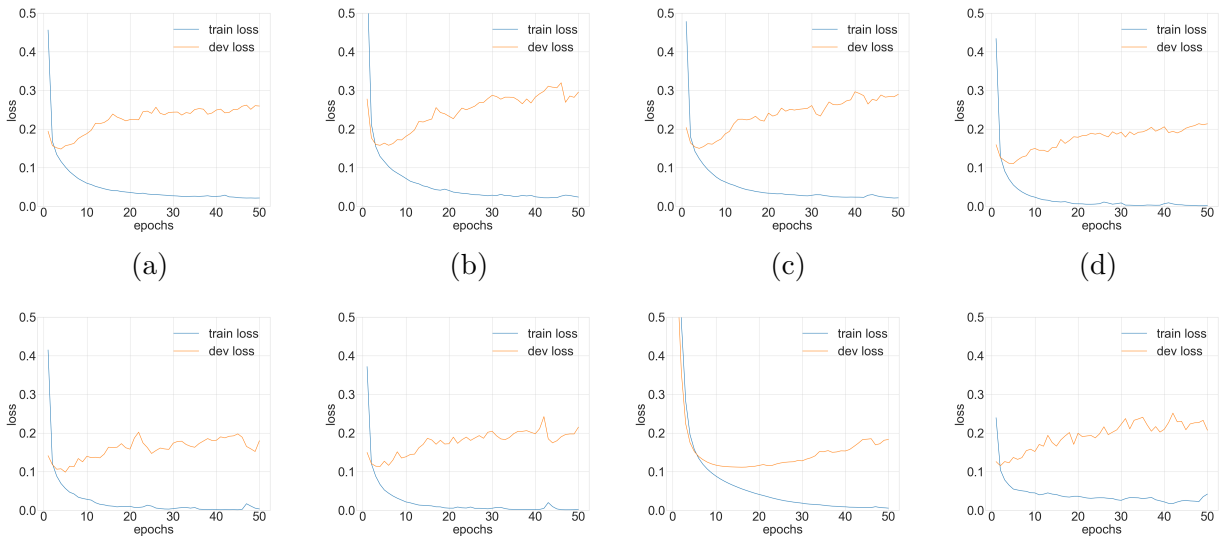


Figure 2: LSTM with Adam and cross entropy loss