Homework #1 Report

Applied Deep Learning

資工碩一　張凱庭　R10922178

---

1. **Data processing**

   For the data processing part, I directly use the sample code provided by TA. It first split the corpus by space and built a **vocab** set by collecting all the "tokens" appearing in the train/eval set. Then, the GloVe pre-trained word vectors(glove.840B.300d.zip) are used for the embedding vectors. In this pre-trained model, each word was represented by 300d vectors. For the non matching tokens, their embedding vectors were randomly created.

	Intent classification	Slot tagging
Unique token	6491	4117
match with **GloVe**	5435	3000
Coverage	83.73%	72.86%

2. **Describe your intent classification model**

Model architecture

```
SeqClassifier(
  (embed): Embedding(6491, 300)
  (lstm): LSTM(300, 64, batch_first=True, bidirectional=True)
  (linear): Linear(in_features=256, out_features=64, bias=True)
  (relu): ReLU()
  (dropout): Dropout(p=0.1, inplace=False)
  (out): Linear(in_features=64, out_features=150, bias=True)
)
```

This model was mainly combined with 3 layer, $Embedding,\ LSTM,\ Linear$ layer and can be formulated as below:

$$W = Embedding(S)$$

$$h_t,\ c_t = LSTM(w_t,\ h_{t-1},\ c_{t-1})$$

$$P = Linear_{out}(ReLU(Linear(avgpool(H) \oplus maxpool(H))))$$

$$y_{pred\_cls} = argmax(P)$$

S : The input sentence where each word is converted to word index, and each sentence was padded to the same length of 128.

w_t: The t-th word vector in a sentence which is a 300 length vector.

W: $[w_0, w_1, \dots, w_t]$

h_t: hidden state of $LSTM$ layer at timestamp t.

H : $[h_0, h_1, \dots, h_t]$

The concatenation result of average pooling and max pooling of H was then projected to the dimension as the number of classes by the $Linear$ layer.

Training Setting	
Epochs	20
Loss funciton	Cross Entropy Loss
Optimizer	Adam
Learning rate	0.001
Batch size	64

Model Performance	
Accuracy	0.89022

3. **Describe your slot tagging model**

Model architecture

```
SlotClassifier(
  (embed): Embedding(4118, 300)
  (emb_ln): LayerNorm((300,), eps=1e-05, elementwise_affine=True)
  (rnn): LSTM(300, 512, num_layers=2, batch_first=True, bidirectional=True)
  (lstm_ln): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
  (softmax): Softmax(dim=1)
  (dropout): Dropout(p=0.1, inplace=False)
  (out): Linear(in_features=1024, out_features=9, bias=True)
)
```

This model was mainly combined with 3 layer, $Embedding$, $LSTM$, $Linear$ layer and can be formulated as below:

$$W = LayerNorm(Embedding(S))$$

$$h_t, c_t = LSTM(w_t, h_{t-1}, c_{t-1})$$

$$P = Linear_{out}(Softmax(Layernorm(H)))$$

$$y_{pred\_cls} = argmax(P)$$

S : The input sentence where each word is converted to word index, and each sentence was padded to the same length of 35.

w_t: The t-th word vector in a sentence which is a 300 length vector.

W: $[w_0, w_1, \dots, w_t]$

h_t: hidden state of $LSTM$ layer at timestamp t.

$H : [h_0, h_1, \dots , h_t]$

P: $[p_1 , p_2 , \dots , p_i]$

p_i: the i-th slot’s probability of each tag, which is a 9d vector.

Training Setting	
Epochs	30
Loss funciton	Cross Entropy Loss
Optimizer	Adam
Learning rate	0.001
Batch size	64

Model Performance	
Joint Accuracy	0.79731

4. Sequence Tagging Evaluation

Seqeval reuslt

```
              precision    recall  f1-score   support

        date       0.74      0.75      0.74       206
  first_name       0.92      0.91      0.92       102
   last_name       0.87      0.79      0.83        78
      people       0.75      0.75      0.75       238
        time       0.84      0.86      0.85       218

   micro avg       0.80      0.80      0.80       842
   macro avg       0.83      0.81      0.82       842
weighted avg       0.80      0.80      0.80       842
```

	Joint Accuracy	Token Accuracy
Eval set	0.8160	0.9565

The joint accuracy weighs each sequence equally, hecne a sequence is considered correct when all the slots were correct. The token accuracy weighs each token equally, it calculates token wise accuracy.

For the evaluation method in seqeval, the “o tag” was ignored, only considering the rest of the tags. The difference between macro and micro average is that macro weighs each class equally whereas micro weighs each sample equally. For the precision metric it calculated the ratio of the correct sample to all the positive samples that we predicted. The recall metric calculated the ratio among all true positive samples how many of them were predicted positive. F1-score is the harmonic average of precision and recall.

5. **Compare with different configurations**
    a. add $layernorm$ after $Embedding$ and $LSTM$

	w/o layernorm	w/ layernorm
Eval set	0.7980	0.8160

Adding $layernorm$ not only improve the accuracy, but also improve the speed of convergence. Without $layernorm$, it takes 25 epoch for the model to converge. Now with $layernorm$, the model only takes 10 epochs to converge.

    b. different activation function

	ReLU	Softmax
Eval set	0.8010	0.8160
Kaggle public	0.7882	0.7973

Changing the activation function from $ReLU$ to $Softmax$ between $Embedding$ and $Linear$ layer improves the model performance, probably because of that $Softmax$ is better in probabilities calculation in this task.

    c. different hidden size of $LSTM$ layer

	64	128	256	512	1024
Eval set	0.7510	0.7920	0.8040	0.8160	0.8170

Increase the hidden size of $LSTM$ layer seems to have a good effect on the accuracy, but has some limits and also increase training time.