Report - ADL HW1

Q1: Data processing

- Describe how do you use the data for intent_cls.sh, slot_tag.sh:
  - a. How do you tokenize the data.
    - The preprocess in preprocess_intent.py and preprocess_slot.py. It will split all the sequences to the bag of words, and then store the words with the descending frequency. That is, the most common word will get lowest index. And add the token 'PAD' for padding and 'UNK' unknown into the bag of tokens.
  - b. The pre-trained embedding you used.
    - The embedding from preprocess_intent.py and preprocess_slot.py. Using glove.
- If you use the sample code, you will need to explain what it does in your own ways to answer Q1.

Q2: Describe your intent classification model.

Describe

- a. your model
  - model
    - GRU
    - ```
      SeqClassifier(
        (embed): Embedding(6491, 300)
        (gru): GRU(300, 512, num_layers=2, batch_first=True, dropout=0.1,
      bidirectional=True)
        (classifier): Sequential(
          (0): Dropout(p=0.1, inplace=False)
          (1): Linear(in_features=1024, out_features=150, bias=True)
        )
        (batchnorm): BatchNorm1d(1024, eps=1e-05, momentum=0.1,
      affine=True, track_running_stats=True)
      )
      ```
  - forward
    - get text [batch_size, seq_len]
    - feed into embedding [batch_size, seq_len, embed_dim]
    - feed embedding into gru and get encode [batch_size, seq_len, hid_dim * n_dir]
    - batchnorm to [batch_size, hid_dim * n_dir]
    - feed into classifier to get logits [batch_size, num_class]
- b. performance of your model. (public score on kaggle)
  - 0.92533
- c. the loss function you used.
  - Cross Entropy
    $$-\sum\limits_{x}{p(x) * logq(x)}$$
- d. The optimization algorithm (e.g. Adam), learning rate and batch size.
      ```

    - optimization algorithm: AdamW
    - learning rate: 0.0014
    - weight decay: 0.14
    - batch size: 256

Q3: Describe your slot tagging model.

Describe

- a. your model
    - model
        - GRU
        - ```
          SeqTagger(
            (embed): Embedding(4117, 300)
            (gru): GRU(300, 512, num_layers=2, batch_first=True, dropout=0.1,
          bidirectional=True)
            (classifier): Sequential(
              (0): Dropout(p=0.1, inplace=False)
              (1): Linear(in_features=1024, out_features=9, bias=True)
            )
            (batchnorm): BatchNorm1d(1024, eps=1e-05, momentum=0.1,
          affine=True, track_running_stats=True)
          )
          ```
    - forward
        - get tokens [batch_size, seq_len]
        - feed into embedding [batch_size, seq_len, embed_dim]
        - feed embedding into gru and get encode [batch_size, seq_len, hid_dim * n_dir]
        - feed into classifier to get logits [batch_size, seq_len, num_class]
- b. performance of your model. (public score on kaggle)
    - 0.76943
- c. the loss function you used.
    - Cross Entropy with ignore_index
      $$-\sum\limits_{x}{p(x) * logq(x)}$$
- d. The optimization algorithm (e.g. Adam), learning rate and batch size.
    - optimization algorithm: AdamW
    - learning rate: 0.0008
    - weight decay: 0.08
    - batch size: 256

Q4: Sequence Tagging Evaluation

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.

- classification_report

	precision	recall	f1-score	support
          ```

	precision	recall	f1-score	support
date	0.78	0.76	0.77	206
first_name	0.94	0.87	0.90	102
last_name	0.84	0.73	0.78	78
people	0.70	0.68	0.69	238
time	0.83	0.84	0.84	218
micro avg	0.80	0.77	0.78	842
macro avg	0.82	0.78	0.80	842
weighted avg	0.80	0.77	0.78	842

- Token accuracy
  7606 / 7891 = 0.9638829045748321
- Joint accuracy
  791 / 1000 = 0.791
- Comparison
  - According to the classification report, the model does a better job on 'first_name' and does a worse job on 'people'. If you look into the data with more detail, you'll find that unseen word would be guessed as 'first_name' and it exactly belongs to that case. However, when 'people' appears following the number, sometimes it belongs to 'people' but sometimes it belongs to 'O', and it make it more difficult to classify.
  - Since the condtion of joint accuracy is stricter than the token accuracy, the token accuracy is much better than joint accuracy.

Q5: Compare with different configurations

- Please try to improve your baseline method (in Q2 or Q3) with different configuration (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 / ...

hidden: hidden_size
layer: num_layers
lr: learning rate
wd: weight decay
batch: batch size

Model Config (*intent classification*)	**val loss**
hidden=1024, layer=4, lr=0.001, wd=0.01, batch=1024	0.0004
hidden=512, layer=2, lr=0.0025, wd=0.2, batch=512	0.0005
hidden=512, layer=2, lr=0.002, wd=0.2, batch=256	0.0010
hidden=512, layer=2, lr=0.0005, wd=0.05, batch=256	0.0011

Model Config (*slot tagging*)	**val loss**
hidden=1024, layer=4, lr=0.001, wd=0.01, batch=1024	0.0009
hidden=512, layer=2, lr=0.0009, wd=0.09, batch=256	0.0005

The more hidden state is and the deeper GRU goes, the performance won't be better each time because of over-fitting.

- Some possible BONUS tricks that you can try: multi-tasking, few-shot learning, zero-shot learning, CRF, CNN-BiLSTM
- This question will be grade by the completeness of your experiments and your findings.