ADL HW2 Report

Q1:Data processing

1. Tokenizer:
   - The algorithm used in BertTokenizer is “WordPiece”. It is the subword tokenization algorithm and is very similar to BPE. WordPiece first initializes the vocabulary to include every character present in the training data and progressively learns a given number of merge rules. In contrast to BPE, WordPiece does not choose the most frequent symbol pair, but the one that maximizes the likelihood of the training data once added to the vocabulary.

2. Answer span:
   a. When tokenize the question and context, I set “return_offsets_mapping” to TRUE which returns (char_start, char_end) for each token. Then, find start token index and end token index of the current span in the context by “sequence_ids” (indicate what is the context, question, and padding). Finally, use while-loop and offsets to find the start and end positions.
      - While token_start_index < len(offsets) and offsets[token_start_index][0] <= start_char: token_start_index += 1. (Final start= token_start_index-1)
      - while offsets[token_end_index][1] >= end_char: token_end_index -= 1. (Final end= token_end_index+1)
   b. First, use argsort to find start index and end index from output.start_logits and output.end_logits. Then, determine the final start/end position by using offsets mapping. (start_char = offset_mapping[start_index][0], end_char = offset_mapping[end_index][1])

Q2:Modeling with BERTs and their variants

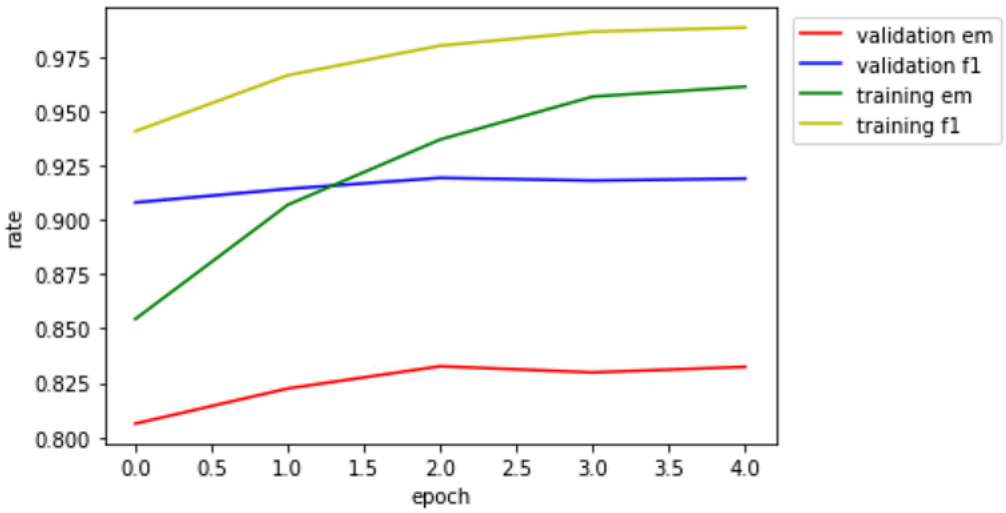
1. Describe
   a. model: “bert-base-chinese”
   b. performance: EM:0.796, F1:0.896 (For QA)

c. loss function: CrossEntropyLoss
d. optimization algorithm: AdamW, learning rate: 1e-5, batch size: 16.

2. Try another type of pretrained model and describe
   a. model: ”hfl/chinese-macbert-base”
   b. performance: EM:0.833, F1:0.919 (For QA)
   c. difference between pretrained model: MacBERT is an improved BERT with novel MLM as correction pre-training task, which mitigates the discrepancy of pre-training and fine-tuning. Instead of masking with [MASK] token, they propose to use similar words which is obtained by using Synonyms toolkit based on word2vec similarity calculations for the masking purpose. If an N-gram is selected to mask, it will find similar words individually. In rare cases, when there is no similar word, they will degrade to use random word replacement. Except for the new pre-training task, they also incorporate the Whole Word Masking (WWM), N-gram masking, and Sentence-Order Prediction (SOP). Note that there is no differences in the main neural architecture from original BERT.

Q3:Curves

1. Plot (learning curve of EM and F1)



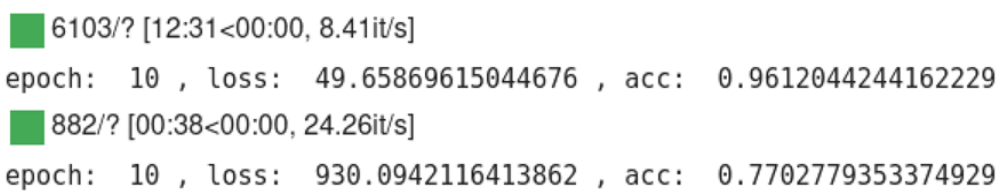
Q4:Pretrained vs Not Pretrained

a. The configuration of the model and how do I train this model: I load the configuration from “bert-base-chinese" but change the hidden_size to 512, the number of attention head to 8, and the number of hidden layers to 6. I use the same training script as pertained BERT model for this problem.

b. the performance of this model v.s. BERT:

- Train from scratch: The optimizer I used is AdamW, learning rate is 1e-4, and the number of epoch is 30. The picture below is the final result. The result shows that we can get good performance on training set but poor performance on validation set. (The first line is training loss and accuracy, the second line is validation loss and accuracy.) (The upper picture below). However, we can get good performance on both training set and validation set when using BERT. (The lower picture below)

```
loss:  136.17594998884306 , acc:  0.908725931995084
loss:  2097.9780416488647 , acc:  0.048496880317640385
```



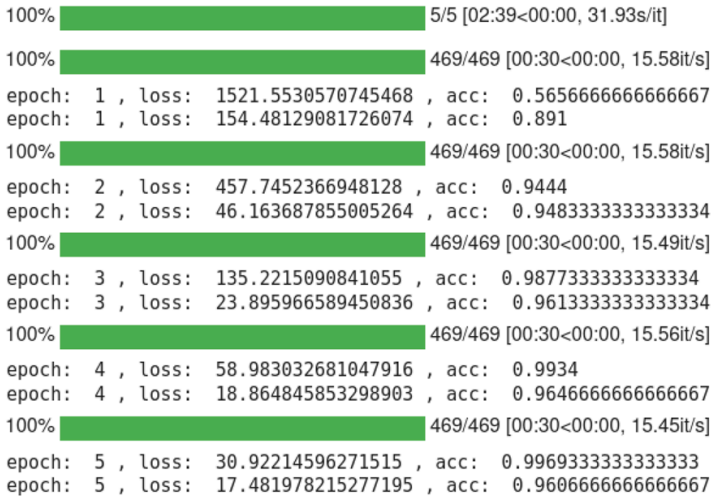
Q5:HW1 with BERTs

a. model:

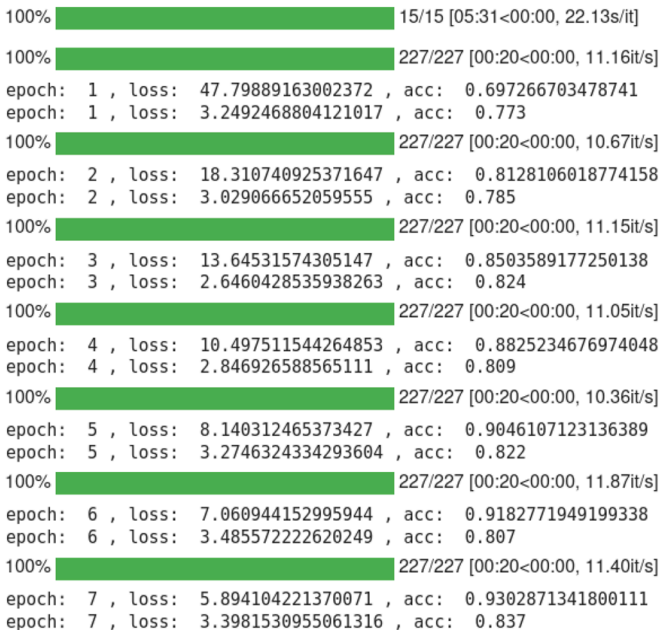
- Intent classification: I use the pertained model “bert-base-uncased”. And I utilize “BertForSequenceClassification” in huggingface to do this case.
- Slot tagging: I use the pertained model “bert-base-uncased”. And I utilize “BertForTokenClassification” in huggingface to do this case.

b. performance:

- Intent classification: In HW1, I used GRU to train the model. The public score on Kaggle is 0.912, the private score is 0.910. The picture below is the model trained on bert. The time consuming is lower than using RNN model and we can get accuracy of 0.96 on evaluation which is higher than using RNN model.(The second line of “epoch:” is validation loss and accuracy.)



- Slot tagging: In HW1, I used GRU to train the model. The public score on Kaggle is 0.783, the private score is 0.786. The picture below is the model trained on bert. The time consuming is lower than using RNN model and we can get accuracy of 0.837 on evaluation which is higher than using RNN model.(The second line of “epoch:” is validation loss and accuracy.)



c. loss function:
   - Intent classification: The loss function used in "BertForSequenceClassification" is CrossEntropyLoss.
   - Slot tagging: The loss function used in "BertForTokenClassification" is CrossEntropyLoss.

d. Optimization algorithm, learning rate and batch size.
   - Intent classification: The optimizer I used is AdamW from huggingface, learning rate is 3e-5, and batch size is 32.
   - Slot tagging: The optimizer I used is AdamW from huggingface, learning rate is 3e-5, and batch size is 32.