ADL HW1 Report

B10705005 陳思如

Q1 Data processing

Tokenizer

- multiple choice

  I use xlnet pretrained model to train the paragraph selection model. The tokenizer I used is the default one from xlnet. The tokenizer is based on **sentence piece.** It takes the input as a long sentence. First, the white space is treated as basic symbol '_'. Then, it cuts the sentences into chunks that it recognizes in its own vocab sets. Then it finally maps those chunks of sentences into the known **sub-word** index.

- question answering

  I use roberta pretrained model to train the question answering model. The default tokenizer from roberta is a derivative from GPT-2. It uses the **byte-pair** encoding level technics. It also uses the sentence piece to treat the white space as a basic symbol. First it transforms all the inputs including white spaces into bytes string. Then it takes common consecutive bytes together then changes them into one byte that can represented them. It aims on putting together the common words and breaking down the rare words to finish the **sentence-piece sub-word** tokenization work.

Answer span

- convert original start/end position on token

  First, we get the offset mapping of the tokens to the original input character position after the tokenization. Second, we check on the sequence id to see which position starts to **mark differently**(0->1 or 1->0) on tokens from the front and check where the token ends (1->0 or 0->1) from the back. We then take this two position as the biggest answer span. Third, we check the previous answer span with our original provided start and end position with our offset map to see if it is out of the original range. If it is, then we set the positions to cls_index. If not, which is our expected result, we then shorten the answer span with continuously checking on the offset map of the original position. Finally, we can get our corresponding answer span by **checking it in the for loop with the offset map**.

- convert predict answer span to final position

  Prediction from the model gives us the start and end position of the answer based on the token. For each input question we may have multiple possible answers that the model gives us. We loops through all the possible ones and maps the start/end position with the offset mapping(map from example to token). Using the map to check if the position is out of range or not. We don't consider the case that is out of range, start position is behind of the end position, or the answer span doesn't reach the context maximum. After we store the all possible answer position, we sort the possible answer with the **softmax score**. Then we take the highest one and map it back to the original context by checking the position on offset map one by one.

Q2

My model

- paragraph selection:
    - pretrained model: hfl/chinese-xlnet-base
    - performance: **eval_metric accuracy 0.964** on valid dataset
    - loss function: default loss function (CrossEntropyLoss)
    - optimizer algorithm: AdamW
    - learining rate 3e-5
    - lr_scedular_type: polynomial
    - batch_size: 1 (1*1)
    - num_warmup_steps 100
    - epoch 1
- question answering
    - pretrained model: hfl/chinese-roberta-wwm-ext
    - performance: **exact_match 80.02** on valid dataset
    - loss function: default loss function (CrossEntropyLoss)
    - optimizer algorithm: AdamW
    - learining rate 3e-5
    - lr_scedular_type: polynomial
    - batch_size: 1 (1*1)
    - num_warmup_steps 20
    - epoch 3

Another type

- paragraph selection & question answering:
    - pretrained model: bert-base-chinese
    - performance: selection eval_metric accuracy 0.948, answering exact_match 74.41
    - loss function: default loss function (CrossEntropyLoss)
    - optimizer algorithm: AdamW
    - learining rate 3e-5
    - lr_scedular_type: linear
    - batch_size: 2 (1*2)
- difference:
    - Bert vs Xlnet:

      Bert is Autoencoding, Xlnet is AutoRegressive. Bert does not consider input in orders but Xlnet does. Then Xlnet learns more dependency pairs and has much denser training signals.
    - Bert vs Roberta:

      Roberta is based on bert but it has more datas and more compute power. Also, Roberta removes the Next sentence prediction in training and adds the dynamic masking into it.

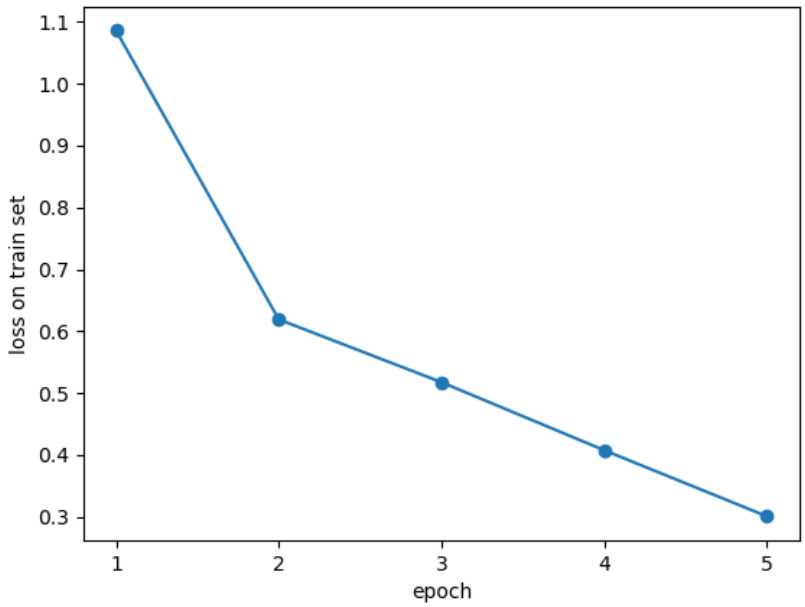
      So we can see that, for the multiple choice and qa cases, the Bert performances is all worse than the other two.

Q3

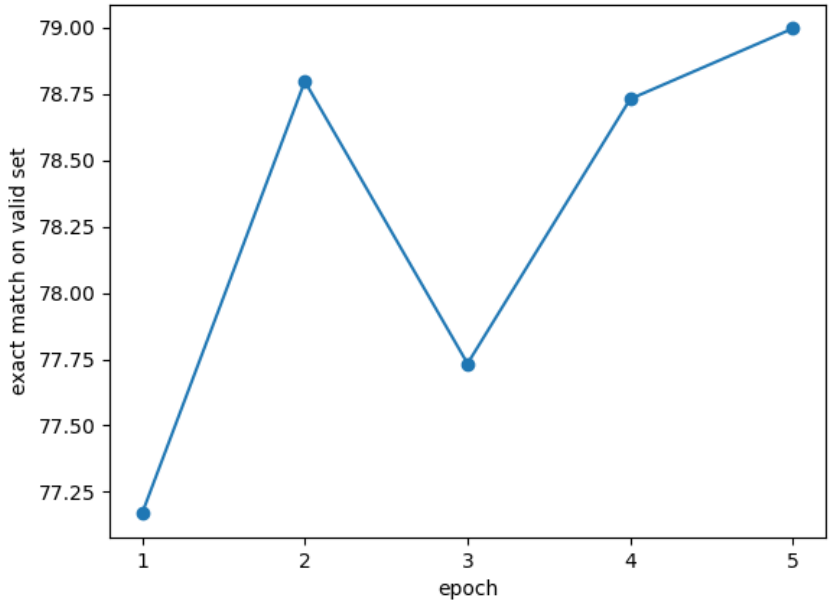
For this question, I train the question answering with different parameters due to the resource limitation.

I changed the epoch to 5, batch size to 2 (1*2)

Loss value learning curve on train set:



Exact Match learning curve on valid set:



Q4

Non-pretrained model for question answering

- configuration:

  The default configuration for bert base model. The tokenizer is the default tokenizer from `hfl/chinese-roberta-wwm-ext`.

  It has

- performance:

  The exact match for valid dataset is only **4.519774011299435**. This is very low compare to Bert and Roberta. It is because the model doesn't have any pretrained knowledge about the words and phrases.

- hyperparameters:

  To check the differences, I use the same set of hyper parameters.

  learning rate: 3e-5

  lr_schedular_type: polynomial

  batch_size: 1 (1*1)

  num_warmup_steps: 100

  epoch: 1