tags: ***ADL***

HW2

```
 python3 run_qa.py --lr=3e-5 --doc_stride=200 --
batch_size=2 --num_epoch=1 --
model_name=hfl/chinese-macbert-large --device=cuda
 python3 qatest.py --lr=3e-5 --doc_stride=200 --
qa_path=./ckpt/macbert.ckpt --
context_file=./data/context.json --
test_file=./data/test.json --pred_file=./pred.csv
--model_name=hfl/chinese-macbert-large --
device=cuda
```

Q1.

1.

I use a bert tokenizer, the bert tokenizer is pretrained on massive dataset. It separates words into wordpieces, and transform them into tokens. My macbert tokenizer uses the traditional Chinese Word Segmentation (CWS) tool to split the text into several words. In this way, we could adopt whole word masking(WWM) in Chinese to mask the word instead of individual Chinese characters

2.

a. I use the `char_to_token` function from tokenizer to convert answer position to token position
b. I first check top-4 probability of start and end position, to see if they are legal indexes and select the combination with largest prob, then I use the `offset_mapping` in tokenizer to know the correspondence of token position and char position.

Q2.

1.

a. I use macbert_large for bert, the config is

```
{
  "architectures": [
    "BertForMaskedLM"
  ],
  "attention_probs_dropout_prob": 0.1,
  "directionality": "bidi",
  "gradient_checkpointing": false,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 1024,
  "initializer_range": 0.02,
  "intermediate_size": 4096,
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 16,
  "num_hidden_layers": 24,
  "pad_token_id": 0,
  "pooler_fc_size": 768,
  "pooler_num_attention_heads": 12,
  "pooler_num_fc_layers": 3,
  "pooler_size_per_head": 128,
  "pooler_type": "first_token_transform",
  "type_vocab_size": 2,
  "vocab_size": 21128
}
```


b. EM

- public: 0.78571
- private: 0.80126

c. cross entropy loss

d. AdamW, lr=3e-5, batch_size=2

2.

a. I used `Langboat/mengzi-bert-base`

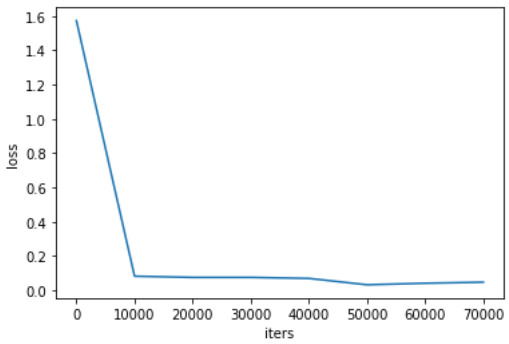
b. EM

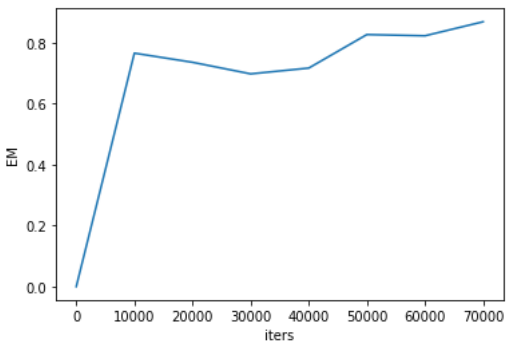
- public: 0.78119
- private: 0.77777

c, d. This model has smaller $hidden\_size, intermediate\_size, num\_hidden\_layers$, and the loss drops slower than macbert_large. And the pretrain masking are different, macbert uses similar words for the masking purpose. A similar word is obtained by using Synonymstoolkit, which is based on word2vec. Mengzi uses LAMB optimizer and mixed-batch training in pretrain stage.

```
{
  "architectures": [
    "BertForMaskedLM"
  ],
  "attention_probs_dropout_prob": 0.1,
  "bos_token_id": 0,
  "directionality": "bidi",
  "eos_token_id": 2,
  "gradient_checkpointing": false,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "output_past": true,
  "pad_token_id": 1,
  "pooler_fc_size": 768,
  "pooler_num_attention_heads": 12,
  "pooler_num_fc_layers": 3,
  "pooler_size_per_head": 128,
  "pooler_type": "first_token_transform",
  "position_embedding_type": "absolute",
  "torch_dtype": "float16",
  "transformers_version": "4.9.2",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
}
```


Q3.





Q4

- config:

```
{
    "architectures": [
      "BertForMaskedLM"
    ],
    "attention_probs_dropout_prob": 0.1,
    "directionality": "bidi",
    "gradient_checkpointing": false,
    "hidden_act": "gelu",
    "hidden_dropout_prob": 0.1,
    "hidden_size": 256,
    "initializer_range": 0.02,
    "intermediate_size": 1024,
    "layer_norm_eps": 1e-12,
    "max_position_embeddings": 512,
    "model_type": "bert",
    "num_attention_heads": 4,
    "num_hidden_layers": 6,
    "pad_token_id": 0,
    "pooler_fc_size": 256,
    "pooler_num_attention_heads": 4,
    "pooler_num_fc_layers": 3,
    "pooler_size_per_head": 128,
    "pooler_type": "first_token_transform",
    "type_vocab_size": 2,
    "vocab_size": 21128
  }
```

I train from a config with smaller layers and sizes, and used the same learning rate as the pretrained scene.

```
configuration = BertConfig.from_pretrained('./macconf.json')
model = BertForQuestionAnswering(configuration).to(device)
```

- EM:0.03

Q5

a.

- Intent: `BertForSequenceClassification` pretrained on `bert-base-uncased`
- Slot: `BertForTokenClassification` pretrained on `bert-base-uncased`

The model was pretrained with two objectives:
Masked language modeling (MLM): taking a sentence, the model randomly masks 15% of the words in the input then run the entire masked sentence through the model and has to predict the masked words. This is different from traditional recurrent neural networks (RNNs) that usually see the words one after the other, or from autoregressive models like GPT which internally mask the future tokens. It allows the model to learn a bidirectional representation of the sentence.
Next sentence prediction (NSP): the models concatenates two masked sentences as inputs during pretraining. Sometimes they correspond to sentences that were next to

each other in the original text, sometimes not. The model then has to predict if the two sentences were following each other or not.

b.

- intent: 0.95288

- slot: 0.82636

c. cross entropy loss

d. optimizer = adam, lr = 3e-5, batch_size = 32