112-1 ADL HW1 Report

`b10902138 陳德維`

Q1: Data processing

- **Tokenizer**:
  - I used the `BertTokenizer` for the `hfl/chinese-roberta-wwm-ext` model.
    1. While it's hard to handle pure Chinese input, it maps each symbol to an unique token id.
       - Special tokens are added:
         - `[CLS]` : Usually placed in the front of the sequence
         - `[SEP]` : Added between context containing 2 or more sequences
         - `[UNK]` : Representing symbols that does not exist in the BERT vocab.
         - `[PAD]` : Used for padding
         - `[MASK]` : Used for word masking
    2. It uses the token mapping and process the input data (in form of "swag", "squad") into following parts:
       - `input_ids` : Tokenizer split the original input data into tokens mapping to the vocab of the tokenizer (we can find the mapping in the `tokenizer.json`), and this list contains the numerical representations of token later used as input of the model.
       - `token_type_ids` : Tells us which part is the question, which part is the context using 0 and 1 representation.
       - `offset_mapping` : It contains intervals of each token's (start , end) position in the original input. (Which helps us finding its original position later)
       - `overflow_to_sample_mapping` : When handling long inputs, truncation may be needed, this list records the mapping from tokenized chunks to the original inputs.
- **Answer Span**:
  - Converting the answer span:
    - Iterate through the `offset_mapping` :
      - 1. Calculate the answer's (start, end) position in the original input by:
        - `start = answers["answer_start"][0]`
        - `end = start + len(answers["text"][0])`
      - 2. Use `token_type_ids` to get the (start, end) position of current context span
      - 3. Check if the answer's position is in the current context span by comparing `offset_mapping` of the current span to the position of the original input with the position in step 1.
        - If yes, we found it.
        - If not, marked the answer to `Impossible` (achieved by setting `[CLS]` index

as answer)

- Rules to determine the final start-end position:
  - I choose the one with highest probability predicted by the model to be the answer.

Q2: Modeling with BERTs and their variants

1. Huggingface QA Model using `bert-base-chinese` as pre-trained LM
   - Performance
     - Evaluation : `'exact_match': 79.02957793286807`
     - Kaggle: `0.74593`
   - Loss function: `The average of the sum of a Cross-Entropy for the start and end positions`
   - Optimization algorithm: `Adam`
   - Effective batch size: `8`
   - Learning rate: `3e-5`
   - Epoch: `3`
2. Huggingface QA Model using `hfl/chinese-roberta-wwm-ext` as pre-trained LM
   - Performance
     - Evaluation : `'exact_match': 81.45563310069791`
     - Kaggle: `0.78028
   - Loss function: `The average of the sum of a Cross-Entropy for the start and end positions`
   - Optimization algorithm: `Adam`
   - Effective batch size: `8`
   - Learning rate: `3e-5`
   - Epoch: `3`

- The difference between the 2 models:

Model	Masking	Training Tokens	Data Source	Performance
bert-base-chinese	WordPiece	0.4B	wiki	0.74593
chinese-roberta-wwm-ext	Whole Word Masking	5.4B	wiki+extension	0.78208

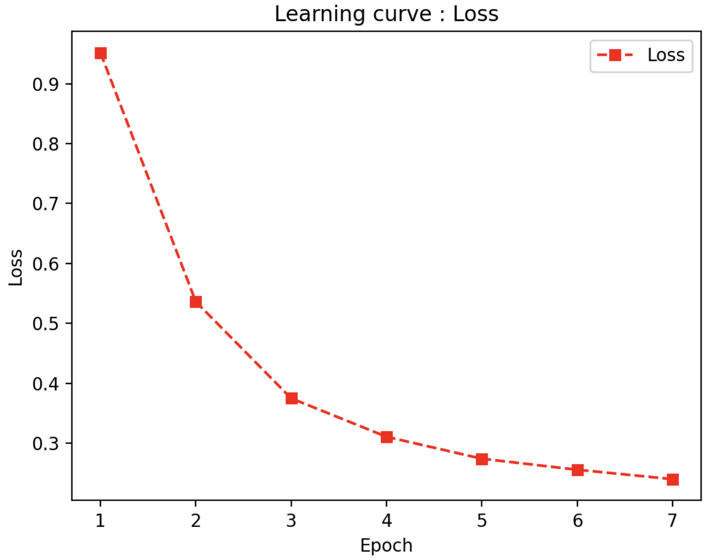
`Performance is compared with same parameters based on Kaggle's public score`

Q3: Curves

- **Model info**:

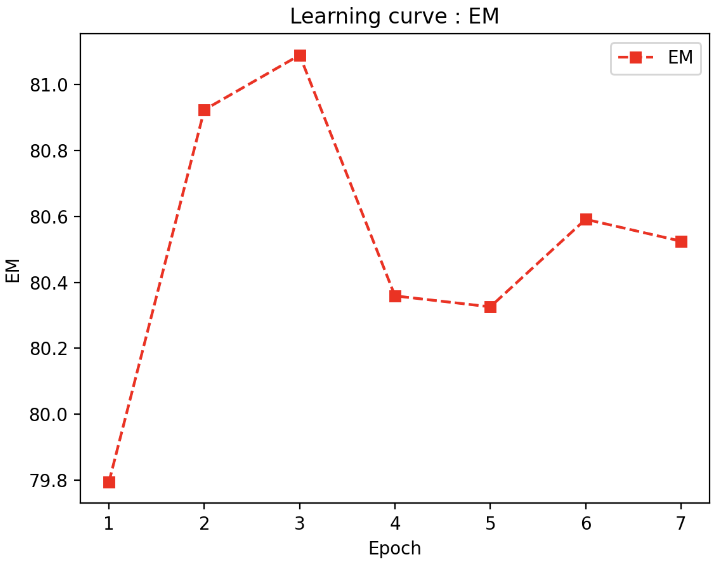
- Huggingface QA Model using `hfl/chinese-roberta-wwm-ext` as pre-trained LM
    - Loss function: `The average of the sum of a Cross-Entropy for the start and end positions`
    - Optimization algorithm: `Adam`
    - Effective batch size: `8`
    - Learning rate: `3e-5`
    - Epoch: `7`
- **Learning curve of the loss value** (rounded off to the 3rd decimal place)

Epoch	1	2	3	4	5	6	7
Loss	0.952	0.536	0.375	0.311	0.273	0.255	0.238



- **Learning curve of the Exact Match metric value** (rounded off to the 3rd decimal place)

Epoch	1	2	3	4	5	6	7
EM	79.794	80.924	81.090	80.359	80.326	80.592	80.525



Q4: Pre-trained v.s. Not Pre-trained

- **Configuration**

```
Model config BertConfig {
  "_name_or_path": "bert-base-chinese",
  "architectures": [
    "BertForMaskedLM"
  ],
  "attention_probs_dropout_prob": 0.1,
  "classifier_dropout": null,
  "directionality": "bidi",
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 384,
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 6,
  "num_hidden_layers": 6,
  "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",
  "position_embedding_type": "absolute",
  "transformers_version": "4.34.0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
}
```

- **Hyper-parameters :

```
# Model from scratch**
python QA.py \
  --tokenizer_name bert-base-chinese \
  --config_name bert-base-chinese \
  --train_file ./part2/train_part2.json \
  --validation_file ./part2/valid_part2.json \
  --max_seq_length 512 \
  --per_device_train_batch_size 4 \
```

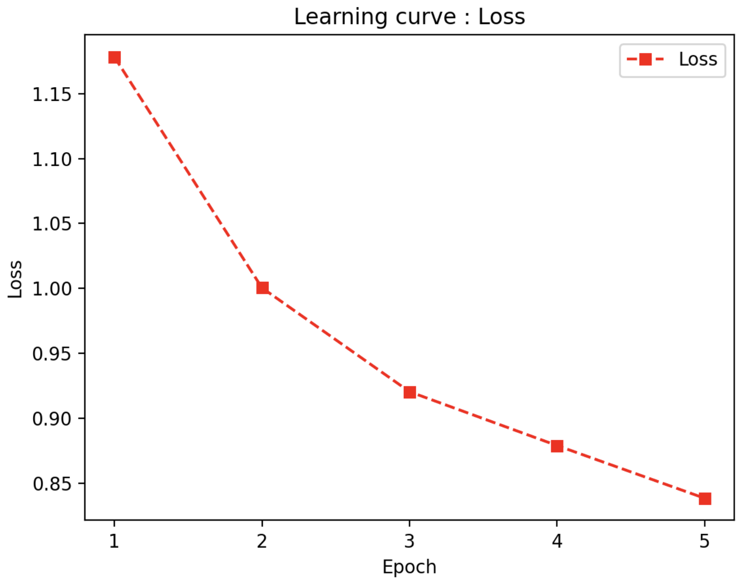
```
  --gradient_accumulation_steps 2 \
  --learning_rate 3e-5 \
  --num_train_epochs 3
# Model with pretrained
python QA.py \
  --model_name_or_path hfl/chinese-roberta-wwm-ext \
  --train_file ./part2/train_part2.json \
  --validation_file ./part2/valid_part2.json \
  --max_seq_length 512 \
  --per_device_train_batch_size 4 \
  --gradient_accumulation_steps 2 \
  --learning_rate 3e-5 \
  --num_train_epochs 3
```

- **Performance**:
  - EM evaluation metric under same hyper-parameters
    - With pre-trained weight: `79.02957793286807`
    - Without pre-trained weight: `6.247922897972749`
  - We can see that its hard to train our model without the pre-trained LM

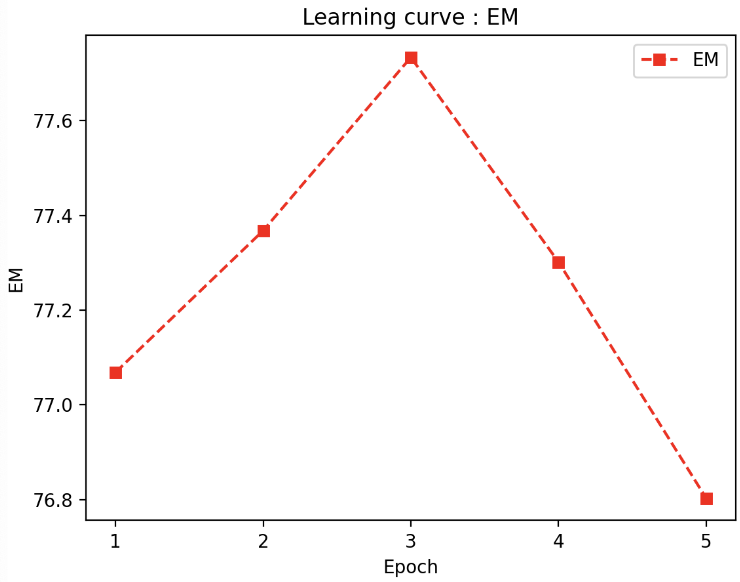
Q5: Bonus

- **Strategy**:
  - Since we want an end-to-end model, I aimed to reuse the phase 2 QA model.
  - For data preprocessing, I concatenate all 4 choices into 1 long context and calculate the relative start_index for the answer based on the "long context", then manage to create the input data in `squad` dataset form to feed the QA model.
- **Model info**:
  - Huggingface QA Model using `hfl/chinese-roberta-wwm-ext` as pre-trained LM
    - Loss function: `The average of the sum of a Cross-Entropy for the start and end positions`
    - Optimization algorithm: `Adam`
    - Effective batch size: `8`
    - Learning rate: `3e-5`
    - Epoch: `5`
- **Performance**:
  - As we can see, the model does learn but in a very slow pace.

Epoch	1	2	3	4	5
Loss	1.178	1.000	0.921	0.879	0.839
EM	77.069	77.368	77.733	77.301	76.803



Learning curve : Loss
Loss
Loss
1.15
1.10
1.05
1.00
0.95
0.90
0.85
1
2
3
4
5
Epoch



Learning curve : EM
EM
EM
77.6
77.4
77.2
77.0
76.8
1
2
3
4
5
Epoch