ADL hw1
R12922121 游誼澐

Q1: Data processing

<u>**Tokenizer:**</u>

Describe in detail about the tokenization algorithm you use. You need to explain what it does in your own ways.

I use the model "hfl/chinese-macbert-large" provided by HuggingFace with the HuggingFace API AutoTokenizer to implement my tokenization algorithm. This algorithm will directly create a instance of the relevant architecture tokenizer class, in my case, "BertTokenizer".

In English, BertTokenizer splits the text into sub-words, the so-called WordPiece tokenization method. However, Chinese words cannot be further divided into subwords, so BertTokenizer splits every individual Chinese word.

If the size of tokenizer is larger than the model's input embeddings, then token embeddings are resized.

<u>**Answer Span:**</u>

1. **How did you convert the answer span start/end position on characters to position on tokens after BERT tokenization?**

   I set the parameter "return_offsets_mapping" to 1 in Tokenizer, specifically the Bert Tokenizer, I used. The Tokenizer includes "offset_mapping", which contains all start and end position of each sub-word.
   Here's the breakdown of the process:
   1. Iterate through the "offset_mapping," which means examining all examples in the dataset and going through the token positions.
   2. Obtain the start and end positions in character data.
   3. Locate the start and end positions in the tokenized data.

2. **After your model predicts the probability of answer span start/end position, what rules did you apply to determine the final start/end position?**

   Do post processing function, which contains below steps:
   (1) For every example,
   (2) Argsort the start indexes and end indexes
   (3) Iterate through all the indexes. If the index is rational, save the offsets (from

the "offset mapping"), score, start and end logit.

(4) Use the offsets to gather the answer in the original context.

Q2: Modeling with BERTs and their variants

o **Your model:**

I employ the hfl/chinese-roberta-wwm-ext pretrained model for multiple-choice tasks, whereas I choose the hfl/chinese-macbert-large pretrained model when working on span selection.

o **The performance of your model:**

a. Multiple choice:

eval_accuracy: 0.9561316051844466

b. Question answering:

eval_exact_match: 83.05084745762711

eval_f1: 83.05084745762711

o **The loss function you used:**

Cross entropy loss

o **The optimization algorithm (e.g. Adam), learning rate and batch size.**

a. Multiple answer:
- Optimizer: AdamW
- Learning rate: 3e-5
- Batch size: 1
- Gradient accumulation: 2

b. Question answering
- Optimizer: AdamW
- Learning rate: 3e-5
- Batch size: 1
- Gradient accumulation: 2

- **Try another type of pre-trained LMs and describe**

o **Your model:**

I compared hfl/chinese-macbert-large and hfl/chinese-roberta-wwm-ext, with all other hyperparameters fixed. The left side is hfl/chinese-macbert-large, and the other side is hfl/chinese-roberta-wwm-ext.

```
ADL_2023_NTU > hw1 > sample_output_mbl_n2 > {} config.json > ...
 1     {
 2         "_name_or_path": "hfl/chinese-macbert-large",
 3         "architectures": [
 4             "BertForQuestionAnswering"
 5         ],
 6         "attention_probs_dropout_prob": 0.1,
 7         "classifier_dropout": null,
 8         "directionality": "bidi",
 9         "gradient_checkpointing": false,
10         "hidden_act": "gelu",
11         "hidden_dropout_prob": 0.1,
12         "hidden_size": 1024,
13         "initializer_range": 0.02,
14         "intermediate_size": 4096,
15         "layer_norm_eps": 1e-12,
16         "max_position_embeddings": 512,
17         "model_type": "bert",
18         "num_attention_heads": 16,
19         "num_hidden_layers": 24,
20         "pad_token_id": 0,
21         "pooler_fc_size": 768,
22         "pooler_num_attention_heads": 12,
23         "pooler_num_fc_layers": 3,
24         "pooler_size_per_head": 128,
25         "pooler_type": "first_token_transform",
26         "position_embedding_type": "absolute",
27         "torch_dtype": "float32",
28         "transformers_version": "4.22.2",
29         "type_vocab_size": 2,
30         "use_cache": true,
31         "vocab_size": 21128
32     }
33
```

```
ADL_2023_NTU > hw1 > output_qa > sample_output_rl_n1 > {} config.json > ...
 1     {
 2         "_name_or_path": "hfl/chinese-roberta-wwm-ext",
 3         "architectures": [
 4             "BertForQuestionAnswering"
 5         ],
 6         "attention_probs_dropout_prob": 0.1,
 7         "bos_token_id": 0,
 8         "classifier_dropout": null,
 9         "directionality": "bidi",
10         "eos_token_id": 2,
11         "hidden_act": "gelu",
12         "hidden_dropout_prob": 0.1,
13         "hidden_size": 768,
14         "initializer_range": 0.02,
15         "intermediate_size": 3072,
16         "layer_norm_eps": 1e-12,
17         "max_position_embeddings": 512,
18         "model_type": "bert",
19         "num_attention_heads": 12,
20         "num_hidden_layers": 12,
21         "output_past": true,
22         "pad_token_id": 0,
23         "pooler_fc_size": 768,
24         "pooler_num_attention_heads": 12,
25         "pooler_num_fc_layers": 3,
26         "pooler_size_per_head": 128,
27         "pooler_type": "first_token_transform",
28         "position_embedding_type": "absolute",
29         "torch_dtype": "float32",
30         "transformers_version": "4.22.2",
31         "type_vocab_size": 2,
32         "use_cache": true,
33         "vocab_size": 21128
34     }
35
```

o   The performance of your model.

(left: hfl/chinese-macbert-large / right: hfl/chinese-roberta-wwm-ext)

```
ADL_2023_NTU > hw1 > sample_output_mbl_n2 > {} all_results.json > ...
 1     {
 2         "eval_exact_match": 83.05084745762711,
 3         "eval_f1": 83.05084745762711
 4     }
```

```
ADL_2023_NTU > hw1 > output_qa > sample_output_rl_n1 > {} all_results.json > ...
 1     {
 2         "eval_exact_match": 80.42539049518112,
 3         "eval_f1": 80.42539049518112
 4     }
```

o   **The difference between pre-trained LMs**

Masking:

The chinese-roberta-wwm-ext model utilizes Whole Word Masking(WWM) while the chinese-macbert-large model (MLM as correction BERT) both whole word masking as well as N-gram masking strategies for selecting candidate tokens for masking.

Model size:

The chinese-macbert-large model has a hidden size of 1024, while chinese-roberta-wwm-ext only has 768. Additionally, the intermediate size of chinese-macbert-large(4096) is larger than chinesse-roberta-wwm-ext(3072).
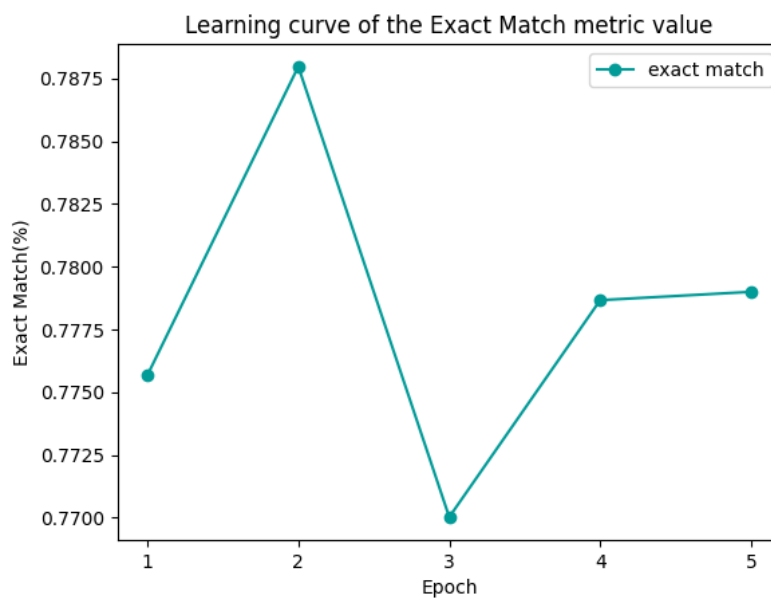
Q3: Curves

Plot the learning curve of your span selection (extractive QA) model. You can plot both the training and validation set or only one set. Please make sure there are at least 5 data points in each curve.

Learning curve of the loss value (0.5%)

(plot on training set)
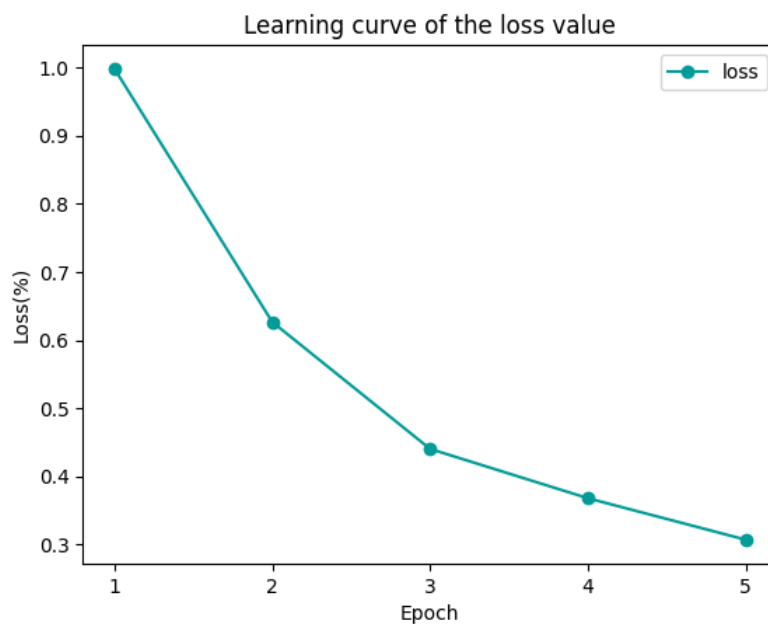
Learning curve of the Exact Match metric value (0.5%)

Q4: Pre-trained vs Not Pre-trained

Train a transformer-based model (you can choose either paragraph selection or span selection) from scratch (i.e. without pretrained weights).
The configuration of the model and how do you train this model (e.g., hyperparameters).

I switched the model for **multiple choice** portion to a non-pretrained model, still using "hfl/chinese-roberta-wwm-ext", with keeping the hyperparameters unchanged. The parameters are stated below:

- max_seq_length 512
- per_device_train_batch_size 1
- gradient_accumulation_steps 2
- learning_rate 3e-5
- num_train_epochs 1

```json
ADL_2023_NTU > hw1 > output > q4-2 > {} config.json > ...
1  {
2    "_name_or_path": "hfl/chinese-roberta-wwm-ext",
3    "architectures": [
4      "BertForMultipleChoice"
5    ],
6    "attention_probs_dropout_prob": 0.1,
7    "bos_token_id": 0,
8    "classifier_dropout": null,
9    "directionality": "bidi",
10   "eos_token_id": 2,
11   "hidden_act": "gelu",
12   "hidden_dropout_prob": 0.1,
13   "hidden_size": 768,
14   "initializer_range": 0.02,
15   "intermediate_size": 3072,
16   "layer_norm_eps": 1e-12,
17   "max_position_embeddings": 512,
18   "model_type": "bert",
19   "num_attention_heads": 12,
20   "num_hidden_layers": 12,
21   "output_past": true,
22   "pad_token_id": 0,
23   "pooler_fc_size": 768,
24   "pooler_num_attention_heads": 12,
25   "pooler_num_fc_layers": 3,
26   "pooler_size_per_head": 128,
27   "pooler_type": "first_token_transform",
28   "position_embedding_type": "absolute",
29   "torch_dtype": "float32",
30   "transformers_version": "4.22.2",
31   "type_vocab_size": 2,
32   "use_cache": true,
33   "vocab_size": 21128
34 }
35
```

- **The performance of this model v.s. BERT.**
  In multiple choice:
  eval_accuracy: 0.021601861083416416
  Compare to the model with pretrained weight, the accuracy decrease of 97.7492% in accuracy from the original model to this model.