# Applied Deep Learning Homework 2

NTUST MBA
M11008019 吳英緩

## Q1. Data processing

### 1. Tokenizer

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

BERT uses tokenization algorithm called WordPiece. It works by splitting words either into the full forms (e.g., one word becomes one token) or into word pieces — where one word can be broken into multiple tokens.

| Word | Token(s) |
|---|---|
| surf | ['surf'] |
| surfing | ['surf', '##ing'] |
| surfboard | ['surf', '##board'] |
| surfboarding | ['surf', '##board', '##ing'] |

WordPiece is the subword tokenization algorithm used for BERT, DistilBERT, and Electra. It's similar to BPE. WordPiece initializes the vocabulary to include every character present in the training data and progressively learns a given number of merge rules at first. The difference between WordPiece and BPE is that WordPiece doesn't 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. How did you convert the answer span start/end position on characters to position on tokens after BERT tokenization?*
*b. After your model predicts the probability of answer span start/end position, what rules did you apply to determine the final start/end position?*

The example script set the parameter 'retrun_offsets_mapping' to True (retrun_offsets_mapping=True) and conducted loop to collect the character start positions and end positions of each token.

The example script checks the logit scores for the n_best start logits and end logits, both of which default to 20 when looking for an answer. Besides, it also excludes positions that the answer is out of the span or with negative length or too long (it has limited the possibilities to 30). The final position is picked the one with the best logit score.

## Q2: Modeling with BERTs and their variants

### 1. BERTs

*a. your model (configuration of the transformer model).*

(a) Context Selection


```
{
  "_name_or_path": "bert-base-chinese",
  "architectures": [
    "BertForMultipleChoice"
  ],
  "attention_probs_dropout_prob": 0.1,
  "classifier_dropout": null,
  "directionality": "bidi",
  "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,
  "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",
  "torch_dtype": "float32",
  "transformers_version": "4.18.0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
}
```


(b) Question Answering


```
{
  "_name_or_path": "bert-base-chinese",
  "architectures": [
    "BertForQuestionAnswering"
  ],
  "attention_probs_dropout_prob": 0.1,
  "classifier_dropout": null,
  "directionality": "bidi",
  "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,
  "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",
  "torch_dtype": "float32",
  "transformers_version": "4.18.0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
}
```


*b. Performance of your model.*

| | Accuracy |
|---|---|
| Context Selection (eval) | 0.95746 |
| Question answering eval EM | 0.78464 |
| Question answering eval F1 | 0.78464 |
| Kaggle Public score | 0.74954 |
| Kaggle Private score | 0.75248 |

*c. The loss function you used.*

Cross entropy loss

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

| | Context Selection | Question Answering |
|---|---|---|
| Optimization Algorithm | Adam | Adam |
| learning rate | 3e-05 | 3e-05 |
| batch size | 3 | 3 |

**2. RoBERTa-wwm-ext**

*a. your model (configuration of the transformer model).*

(a) Context Selection


```
{
  "_name_or_path": "hfl/chinese-roberta-wwm-ext",
  "architectures": [
    "BertForMultipleChoice"
  ],
  "attention_probs_dropout_prob": 0.1,
  "bos_token_id": 0,
  "classifier_dropout": null,
  "directionality": "bidi",
  "eos_token_id": 2,
  "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": 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",
  "torch_dtype": "float32",
  "transformers_version": "4.18.0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
}
```


(b) Question Answering


```
{
  "_name_or_path": "hfl/chinese-roberta-wwm-ext",
  "architectures": [
    "BertForQuestionAnswering"
  ],
  "attention_probs_dropout_prob": 0.1,
  "bos_token_id": 0,
  "classifier_dropout": null,
  "directionality": "bidi",
  "eos_token_id": 2,
  "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": 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",
  "torch_dtype": "float32",
  "transformers_version": "4.18.0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
}
```


*b. Performance of your model.*

| | Accuracy |
|---|---|
| Context Selection (eval) | 0.95812 |
| Question answering eval EM | 0.81820 |
| Question answering eval F1 | 0.81820 |
| Kaggle Public score | 0.78390 |
| Kaggle Private score | 0.78048 |

*c. The loss function you used.*

Cross entropy loss

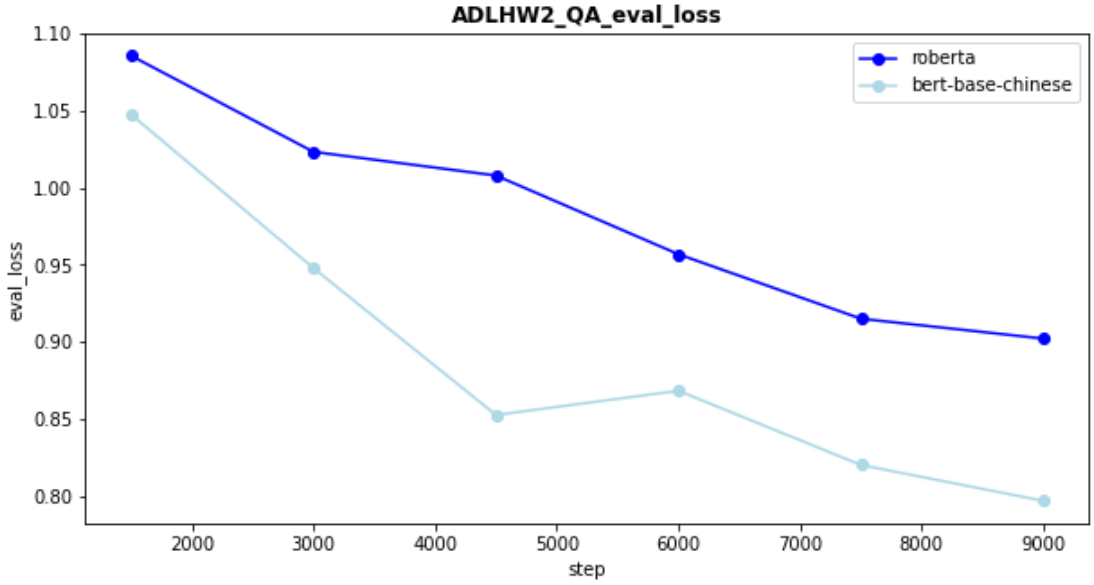
*d. The optimization algorithm (e.g. Adam), learning rate and batch size.*

| | Context Selection | Question answering |
|---|---|---|
| Optimization Algorithm | Adam | Adam |
| learning rate | 3e-05 | 3e-05 |
| batch size | 3 | 3 |

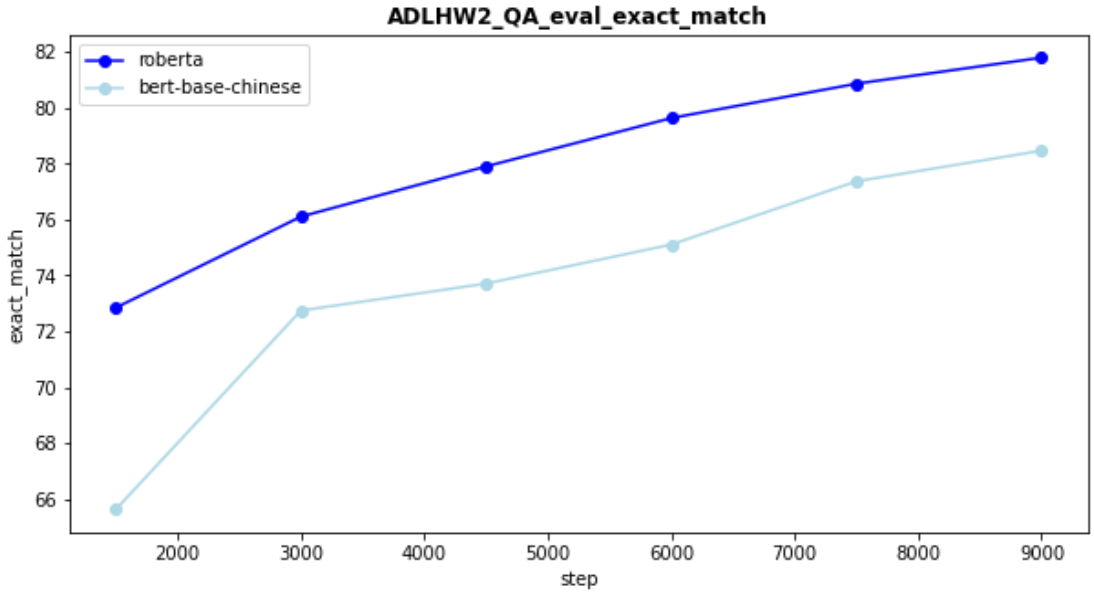
## Q3: Curves

### 1.Plot the learning curve of your QA model

*a. Learning curve of loss.*




*b. Learning curve of EM.*



## Q4: Pretrained vs Not Pretrained

Actually, I chose the Question Answering for this question at first, but the model didn't appear to be training. Therefore, I chose the Multiple Choice afterwards.

*a. Configuration*

I used the same training script without loading the pretrained weights to train the model, so the configuration was the same as the above.


```
{
  "_name_or_path": "bert-base-chinese",
  "architectures": [
    "BertForMultipleChoice"
  ],
  "attention_probs_dropout_prob": 0.1,
  "classifier_dropout": null,
  "directionality": "bidi",
  "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,
  "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",
  "torch_dtype": "float32",
  "transformers_version": "4.18.0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
}
```


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

| | Evaluation Accuracy |
|---|---|
| Context Selection (Not Pretrained) | 0.54403 |
| Context Selection (BERT) | 0.95746 |

Because of the bad evaluation accuracy, I didn't continue training the QA model after the MC training.

Overall, the pretrained weights of BERT were created by hugging face programmers to solve the similar problems, such as multiple choice and question answering. Therefore, it's generally recommended utilizing the pretrained weights to build the model instead of building from scratch.

## Reference

1. huggingface/transformers(github)
https://github.com/huggingface/transformers/tree/main/examples/pytorch
2. Summary of the tokenizers
https://huggingface.co/docs/transformers/tokenizer_summary
3. Question answering
https://huggingface.co/course/chapter7/7?fw=tf
4. How to Build a WordPiece Tokenizer For BERT
https://towardsdatascience.com/how-to-build-a-wordpiece-tokenizer-for-bert-f505d97dddbb