ADL 2022 Fall

HW2

資工碩一 r11922189 邱議禾

Q1: Data processing

1. Tokenizer: Describe in detail about the tokenization algorithm you use. You need to explain what it does in your own ways.
   Using WordPiece Tokenizer.
   WordPiece Tokenizer 一開始將 dataset 內的所有 characters 收集起來，再從此集合中所有 tokens 進行相互配對，找出「兩個 tokens 連續出現的機率」和「兩個 tokens 各自出現機率的乘積」之比值最大的配對，WordPiece Tokenizer 會將符合配對條件的兩個 tokens 相連形成新的 token，不斷尋找配對直到自訂義的停止條件。
2. Answer Span

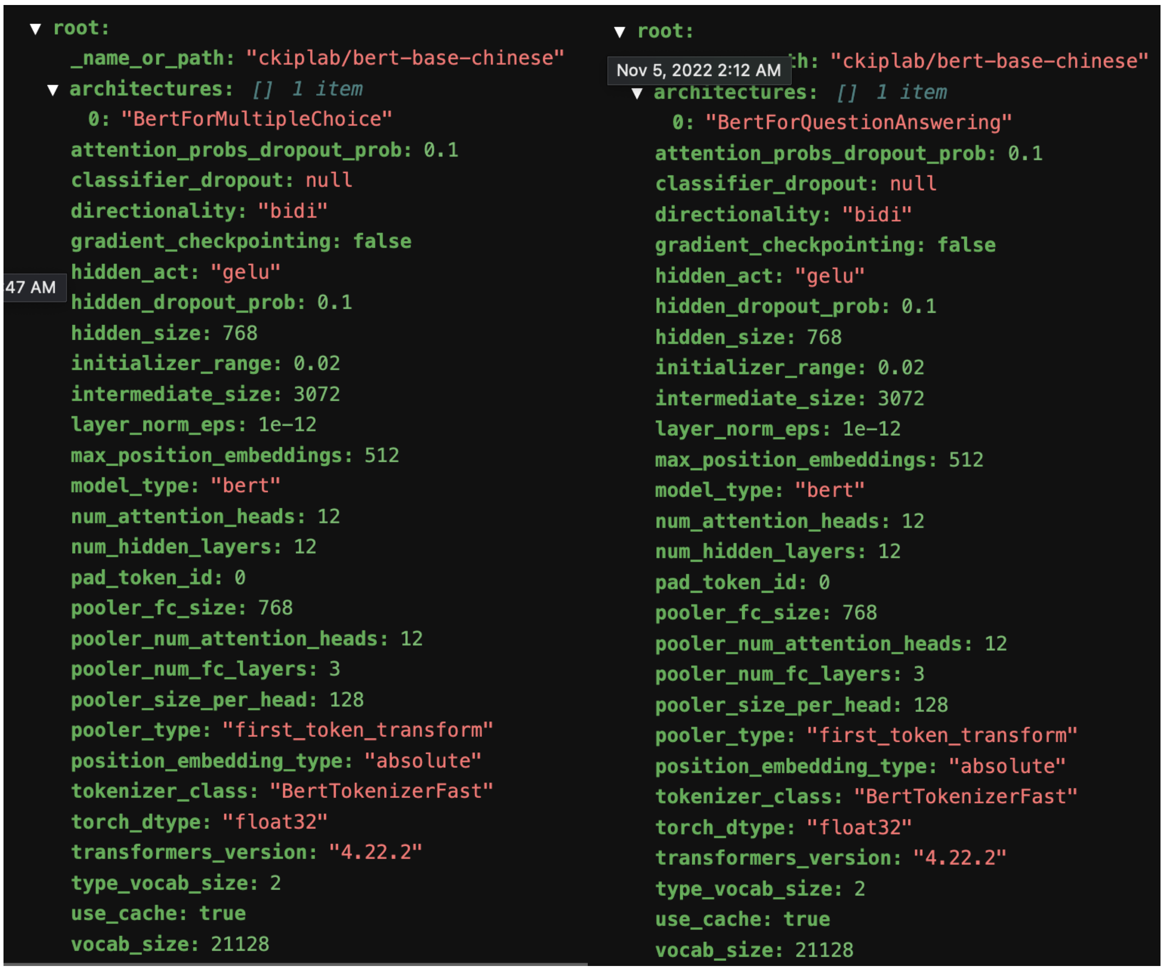
(a) How did you convert the answer span start/end position on characters to position on tokens after BERT tokenization?
First, get context span with offset mapping. Which offers each token its character position range in the original context. The token includes start character position is start token. And the token includes end character position is end token.
Example:
   Context: “An apple a day.”
   Tokens: [“An”, “apple”, “a”, “day”, “ ”, “.”]
   Answer: “An apple”
   Context Span: [(0,2), (2,3), (3,8), (8, 9), (9, 10), (10, 12), (12, 13), (13, 16), (16, 17)]
   Answer Span: [(0,2), (2,3), (3,8)]
   Start Token Position: 0
   End Token Position: 2

(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?
Select a start/end position pair that maximizes start position probability + end position probability. Also, ignore impossible conditions. For example: start position > end position, end position - start position > answer span length.

Q2: Modeling with BERTs and their variants

1. ckiplab/bert-base-chinese

(a) Configuration



```
▼ root:
    _name_or_path: "ckiplab/bert-base-chinese"
  ▼ architectures: [] 1 item
      0: "BertForMultipleChoice"
    attention_probs_dropout_prob: 0.1
    classifier_dropout: null
    directionality: "bidi"
    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
    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"
    tokenizer_class: "BertTokenizerFast"
    torch_dtype: "float32"
    transformers_version: "4.22.2"
    type_vocab_size: 2
    use_cache: true
    vocab_size: 21128
```

```
▼ root:
                        th: "ckiplab/bert-base-chinese"
  ▼ architectures: [] 1 item
      0: "BertForQuestionAnswering"
    attention_probs_dropout_prob: 0.1
    classifier_dropout: null
    directionality: "bidi"
    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
    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"
    tokenizer_class: "BertTokenizerFast"
    torch_dtype: "float32"
    transformers_version: "4.22.2"
    type_vocab_size: 2
    use_cache: true
    vocab_size: 21128
```

Using run_swag.py script for context selection.
https://github.com/huggingface/transformers/blob/main/examples/pytorch/multiple-choice/run_swag.py
Using run_qa.py script for question answering.
https://github.com/huggingface/transformers/blob/main/examples/pytorch/question-answering/run_qa.py
Using bert-base-chinese as pretrained model.
https://huggingface.co/ckiplab/bert-base-chinese

(b) Performance

- Public Score: 0.75226
- Context Selection Accuracy (EM): 0.9541375637054443
- Question Answering Accuracy(EM): 0.7952808241940844

(c) Loss function

- Cross Entropy Loss

(d) Training Arguments

- Context Selection

  Optimization Algorithm: AdamW

  --max_seq_length 512

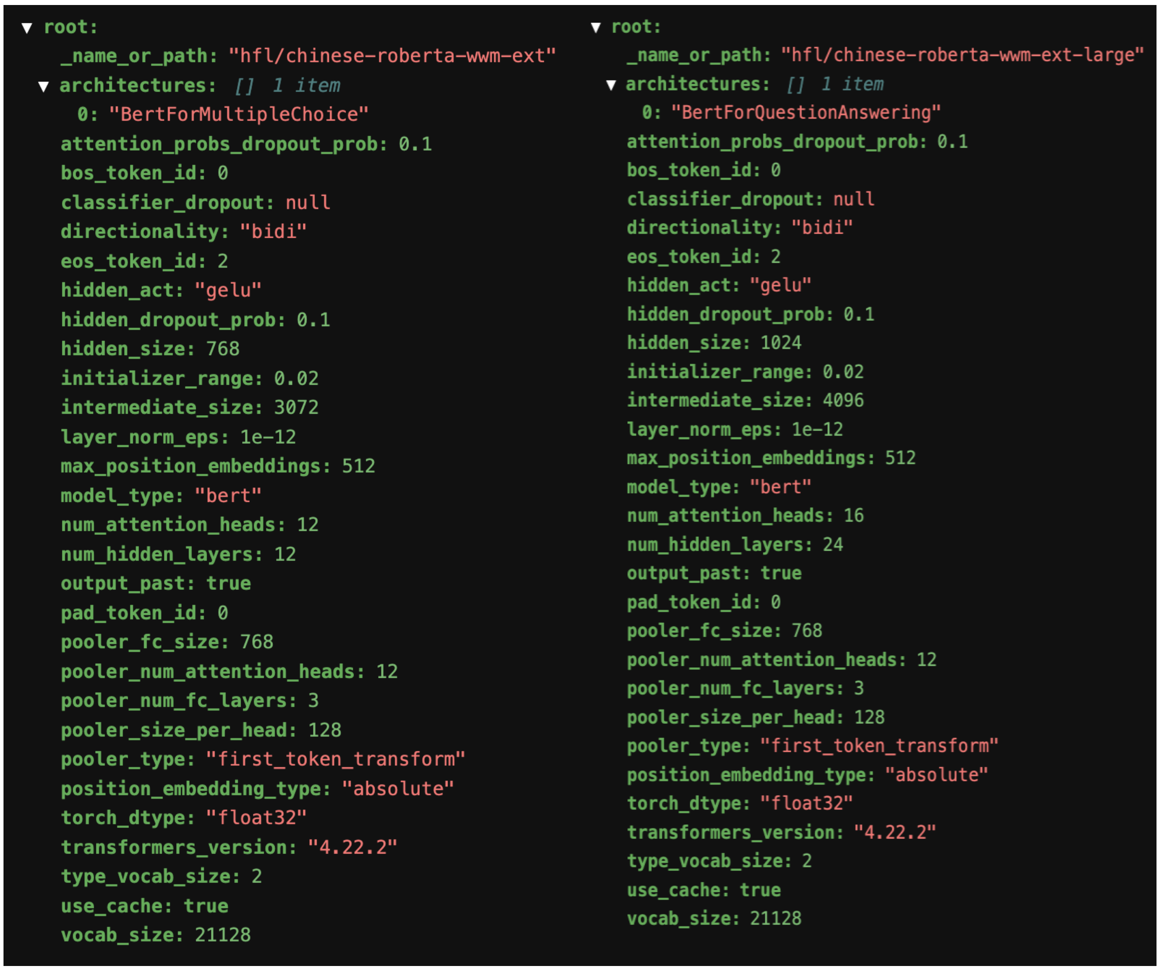
--pad_to_max_length
--evaluation_strategy steps
--eval_steps 1000
--save_steps 1000
--per_device_train_batch_size 1
--per_device_eval_batch_size 1
--gradient_accumulation_steps 2
--learning_rate 3e-5
--num_train_epochs 1
--warmup_ratio 0.1   // Linear Decay
--metric_for_best_model accuracy   // EM
--load_best_model_at_end True

- Question Answering

  Optimization Algorithm: AdamW
  --max_seq_length 512
  --pad_to_max_length
  --evaluation_strategy steps
  --eval_steps 1000
  --save_steps 1000
  --per_device_train_batch_size 1
  --per_device_eval_batch_size 1
  --gradient_accumulation_steps 2
  --learning_rate 3e-5
  --num_train_epochs 3
  --warmup_ratio 0.1   // Linear Decay
  --metric_for_best_model accuracy   // EM
  --load_best_model_at_end True

2. hfl/chinese-roberta-wwm-ext -> hfl/chinese-roberta-wwm-ext-large

(a) Configuration


```
▼ root:
    _name_or_path: "hfl/chinese-roberta-wwm-ext"
  ▼ architectures: [] 1 item
      0: "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: "4.22.2"
    type_vocab_size: 2
    use_cache: true
    vocab_size: 21128
```

```
▼ root:
    _name_or_path: "hfl/chinese-roberta-wwm-ext-large"
  ▼ architectures: [] 1 item
      0: "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: 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
    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: "4.22.2"
    type_vocab_size: 2
    use_cache: true
    vocab_size: 21128
```

Using run_swag.py script for context selection.
https://github.com/huggingface/transformers/blob/main/examples/pytorch/multiple-choice/run_swag.py
Using run_qa.py script for question answering.
https://github.com/huggingface/transformers/blob/main/examples/pytorch/question-answering/run_qa.py
Using hfl/chinese-roberta-wwm-ext as context selection pretrained model.
https://huggingface.co/hfl/chinese-roberta-wwm-ext
Using hfl/chinese-roberta-wwm-ext-large as question answering pretrained model.
https://huggingface.co/hfl/chinese-roberta-wwm-ext-large

(b) Performance
- Public Score: 0.80922
- Context Selection Accuracy (EM): 0.9587903022766113
- Question Answering Accuracy(EM): 0.8384845463609173

(c) Loss Function
- Cross Entropy Loss

(d) Training Arguments
- Context Selection

Optimization Algorithm: AdamW
--max_seq_length 512
--pad_to_max_length
--evaluation_strategy steps
--eval_steps 1000
--save_steps 1000
--per_device_train_batch_size 1
--per_device_eval_batch_size 1
--gradient_accumulation_steps 2
--learning_rate 3e-5
--num_train_epochs 3
--warmup_ratio 0.1   // Linear Decay
--metric_for_best_model accuracy   // EM
--load_best_model_at_end True

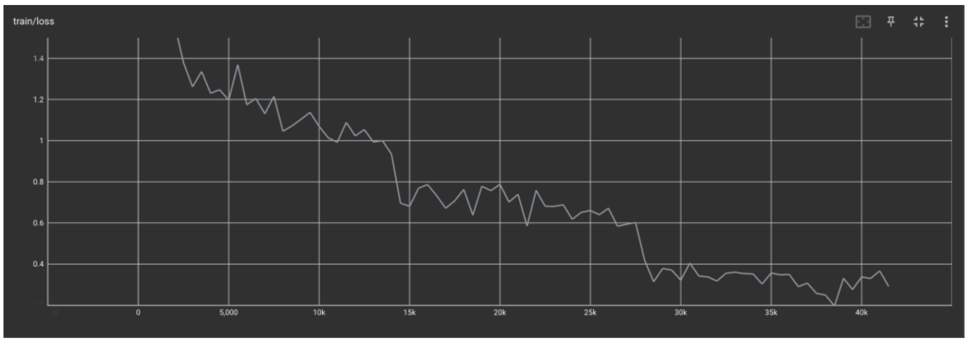
- Question Answering

Optimization Algorithm: AdamW
--max_seq_length 512
--pad_to_max_length
--evaluation_strategy steps
--eval_steps 50
--save_steps 50
--logging_steps 50
--per_device_train_batch_size 8
--per_device_eval_batch_size 8
--gradient_accumulation_steps 8
--eval_accumulation_steps 8
--learning_rate 3e-5
--num_train_epochs 10
--warmup_ratio 0.1   // Linear Decay
--metric_for_best_model accuracy   // EM
--load_best_model_at_end True

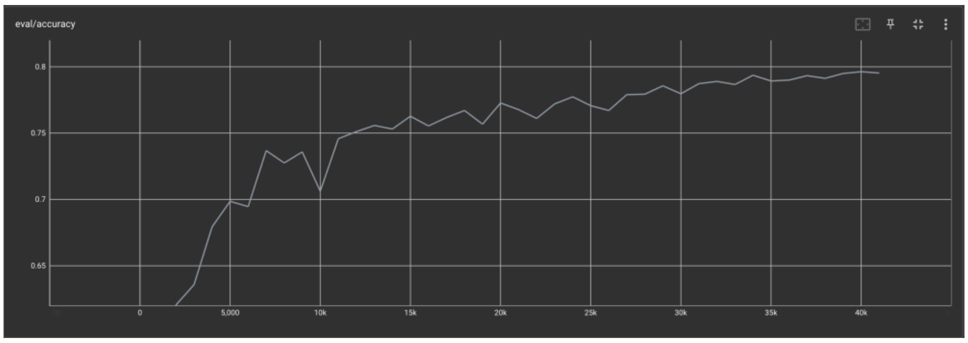
Q3: Curves (QA)

1. ckiplab/bert-base-chinese

(a) Loss



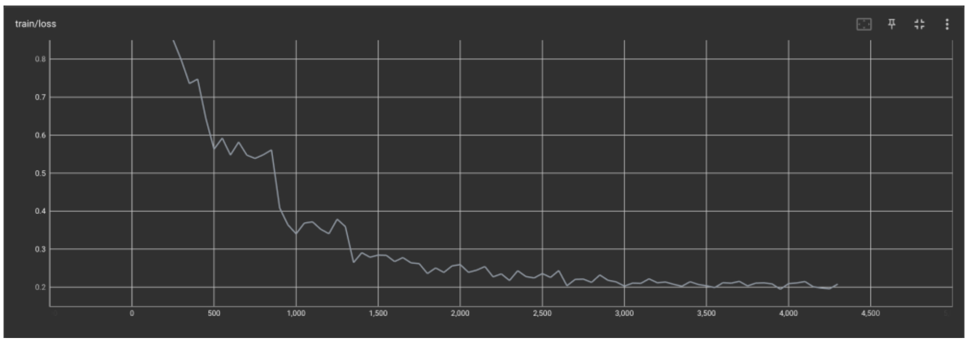
(b) Accuracy (EM)



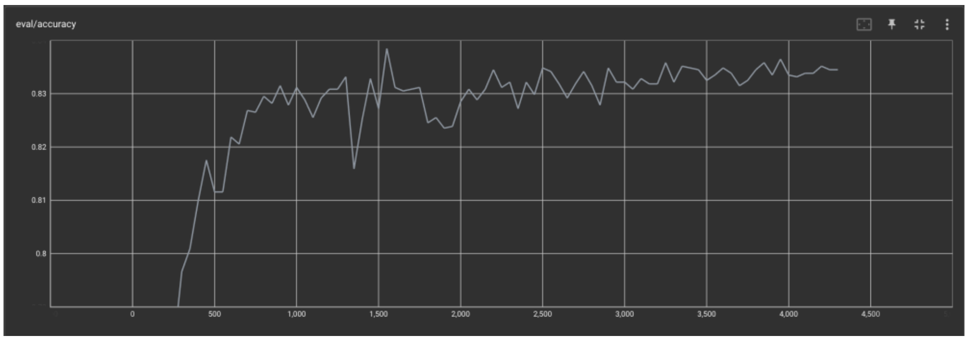
(c)

2. hfl/chinese-roberta-wwm-ext-large

(a) Loss



(b) Accuracy (EM)



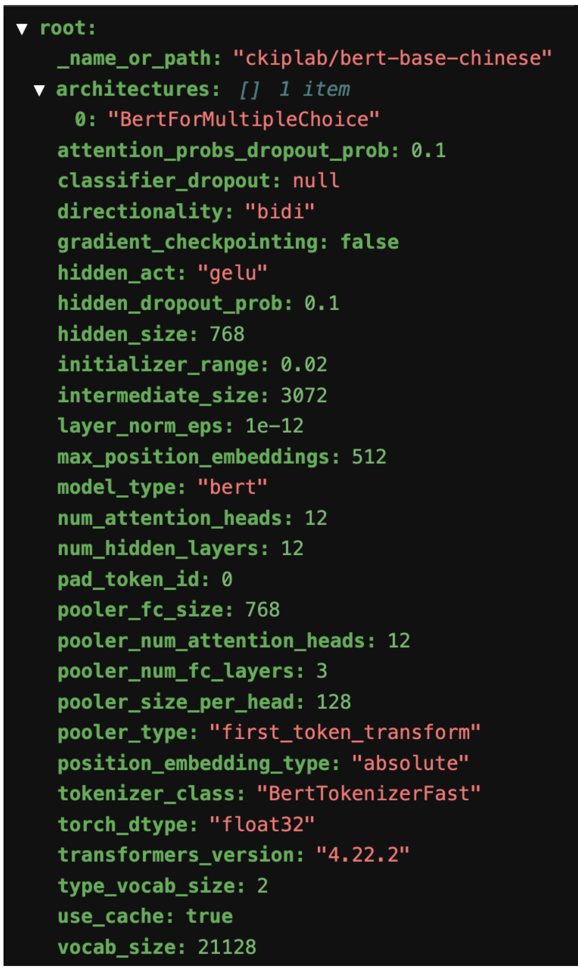
Q4: Pretrained vs Not Pretrained

1. Not Pretrained bert-base-chinese

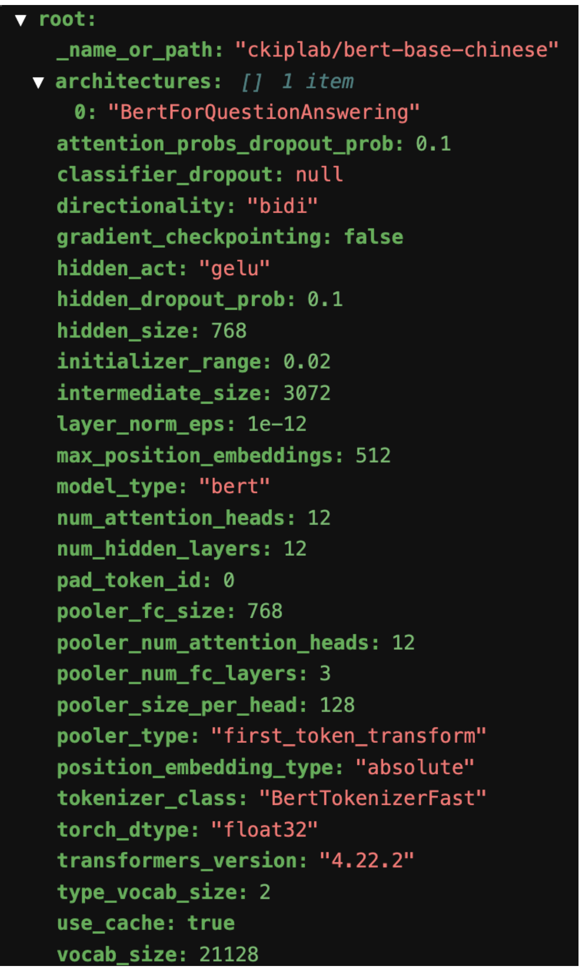
(a) How do I train this model?

All are the same as Q2 bert-base-chinese model. But skip loading pretrained weights.

(b) Configuration



```
▼ root:
   _name_or_path: "ckiplab/bert-base-chinese"
  ▼ architectures: [] 1 item
     0: "BertForMultipleChoice"
   attention_probs_dropout_prob: 0.1
   classifier_dropout: null
   directionality: "bidi"
   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
   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"
   tokenizer_class: "BertTokenizerFast"
   torch_dtype: "float32"
   transformers_version: "4.22.2"
   type_vocab_size: 2
   use_cache: true
   vocab_size: 21128
```



```
▼ root:
   _name_or_path: "ckiplab/bert-base-chinese"
  ▼ architectures: [] 1 item
     0: "BertForQuestionAnswering"
   attention_probs_dropout_prob: 0.1
   classifier_dropout: null
   directionality: "bidi"
   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
   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"
   tokenizer_class: "BertTokenizerFast"
   torch_dtype: "float32"
   transformers_version: "4.22.2"
   type_vocab_size: 2
   use_cache: true
   vocab_size: 21128
```

Using run_swag.py script for context selection.
https://github.com/huggingface/transformers/blob/main/examples/pytorch/multiple-choice/run_swag.py
Using run_qa.py script for question answering.
https://github.com/huggingface/transformers/blob/main/examples/pytorch/question-answering/run_qa.py
Using bert-base-chinese as pretrained model but skip loading pretrained weight.
https://huggingface.co/ckiplab/bert-base-chinese

(c) Loss function

- Cross Entropy Loss

(d) Training Arguments

- Context Selection

  Optimization Algorithm: AdamW
  --max_seq_length 512
  --pad_to_max_length
  --evaluation_strategy steps
  --eval_steps 1000
  --save_steps 1000

--per_device_train_batch_size 1
--per_device_eval_batch_size 1
--gradient_accumulation_steps 2
--learning_rate 3e-5
--num_train_epochs 1
--warmup_ratio 0.1 // Linear Decay
--metric_for_best_model accuracy // EM
--load_best_model_at_end True

- Question Answering

Optimization Algorithm: AdamW
--max_seq_length 512
--pad_to_max_length
--evaluation_strategy steps
--eval_steps 1000
--save_steps 1000
--per_device_train_batch_size 1
--per_device_eval_batch_size 1
--gradient_accumulation_steps 2
--learning_rate 3e-5
--num_train_epochs 3
--warmup_ratio 0.1 // Linear Decay
--metric_for_best_model accuracy // EM
--load_best_model_at_end True

2. The performance of this model v.s. Bert

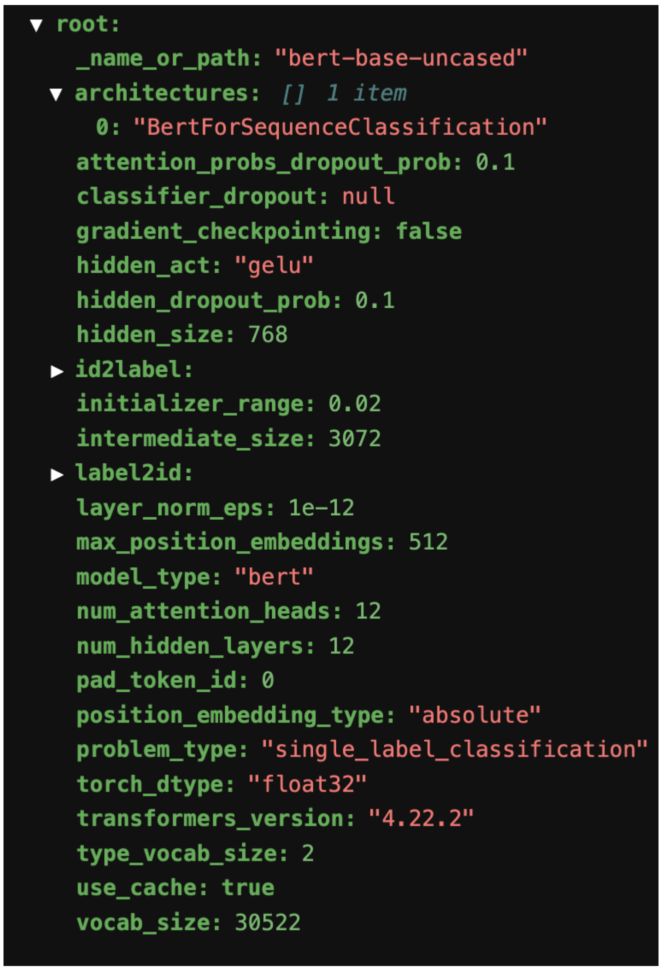
(a) Not Pretrained bert-base-chinese
- Public Score: 0.04339
- Context Selection Accuracy (EM): 0.543702244758606
- Question Answering Accuracy(EM): 0.05982053838484546

(b) Pretrained bert-base-chinese
- Public Score: 0.75226
- Context Selection Accuracy (EM): 0.9541375637054443
- Question Answering Accuracy(EM): 0.7952808241940844

Q5: Bonus: HW1 with BERTs

1. Intent Classification

(a) bert-base-uncased



```
▼ root:
    _name_or_path: "bert-base-uncased"
  ▼ architectures: [] 1 item
      0: "BertForSequenceClassification"
    attention_probs_dropout_prob: 0.1
    classifier_dropout: null
    gradient_checkpointing: false
    hidden_act: "gelu"
    hidden_dropout_prob: 0.1
    hidden_size: 768
  ▶ id2label:
    initializer_range: 0.02
    intermediate_size: 3072
  ▶ label2id:
    layer_norm_eps: 1e-12
    max_position_embeddings: 512
    model_type: "bert"
    num_attention_heads: 12
    num_hidden_layers: 12
    pad_token_id: 0
    position_embedding_type: "absolute"
    problem_type: "single_label_classification"
    torch_dtype: "float32"
    transformers_version: "4.22.2"
    type_vocab_size: 2
    use_cache: true
    vocab_size: 30522
```

Using run_glue.py script.

https://github.com/huggingface/transformers/blob/main/examples/pytorch/text-classification/run_glue.py

Using bert-base-uncased as pretrained model.

https://huggingface.co/bert-base-uncased

(b) Performance

- Public Score: 0.96
- Private Score: 0.95822
- Eval Accuracy: 0.9610000252723694

(c) Loss Function

- Cross Entropy Loss

(d) Training Arguments

Optimization Algorithm: AdamW

--max_seq_length 32

--pad_to_max_length

--evaluation_strategy steps

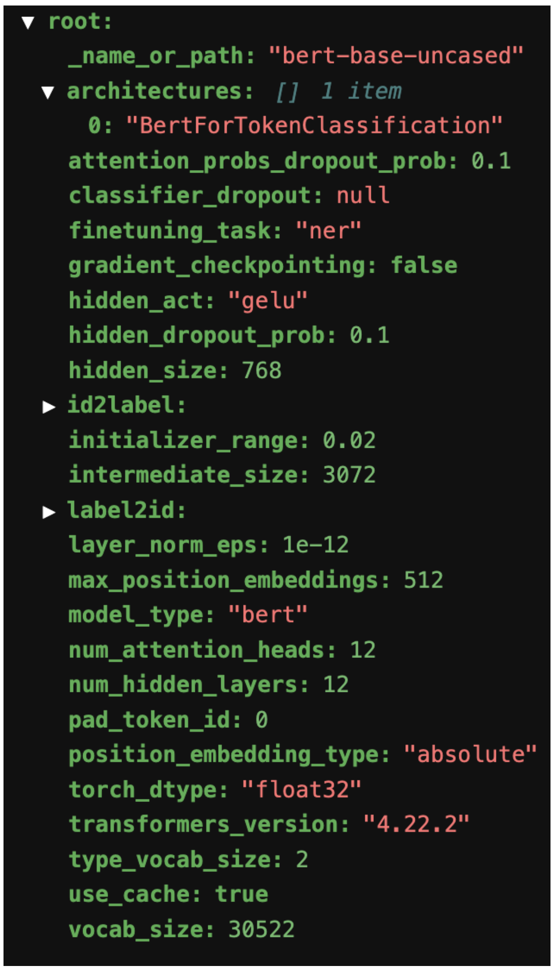
--eval_steps 50

--save_steps 50

```
--logging_steps 50
--per_device_train_batch_size 8
--per_device_eval_batch_size 8
--gradient_accumulation_steps 8
--eval_accumulation_steps 8
--learning_rate 3e-5
--num_train_epochs 5
--warmup_ratio 0.1    // Linear Decay
--metric_for_best_model accuracy
--load_best_model_at_end True
```

2. Slot Tagging

(a) bert-base-uncased



```
▼ root:
    _name_or_path: "bert-base-uncased"
  ▼ architectures: [] 1 item
      0: "BertForTokenClassification"
    attention_probs_dropout_prob: 0.1
    classifier_dropout: null
    finetuning_task: "ner"
    gradient_checkpointing: false
    hidden_act: "gelu"
    hidden_dropout_prob: 0.1
    hidden_size: 768
  ▶ id2label:
    initializer_range: 0.02
    intermediate_size: 3072
  ▶ label2id:
    layer_norm_eps: 1e-12
    max_position_embeddings: 512
    model_type: "bert"
    num_attention_heads: 12
    num_hidden_layers: 12
    pad_token_id: 0
    position_embedding_type: "absolute"
    torch_dtype: "float32"
    transformers_version: "4.22.2"
    type_vocab_size: 2
    use_cache: true
    vocab_size: 30522
```

Using run_ner.py script.

https://github.com/huggingface/transformers/blob/main/examples/pytorch/token-classification/run_ner.py

Using bert-base-uncased as pretrained model.

https://huggingface.co/bert-base-uncased

(b) Performance

- Public Score: 0.80321
- Private Score: 0.82047
- eval_accuracy: 0.9703158695927946
- eval_f1: 0.8172796263864565
- eval_loss: 0.08934387564659119
- eval_precision: 0.8027522935779816
- eval_recall: 0.8323424494649228

(c) Loss Function

- Cross Entropy Loss

(d) Training Arguments

```
Optimization Algorithm: AdamW
--max_seq_length 35
--pad_to_max_length
--evaluation_strategy steps
--eval_steps 50
--save_steps 50
--logging_steps 50
--per_device_train_batch_size 8
--per_device_eval_batch_size 8
--gradient_accumulation_steps 8
--eval_accumulation_steps 8
--learning_rate 3e-5
--num_train_epochs 5
--warmup_ratio 0.1    // Linear Decay
--metric_for_best_model accuracy
--load_best_model_at_end True
```