Q1: Data processing

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

       Bert tokenizer 為 wordpiece，根據 maximum likelihood 選擇 subword。

       演算法如下:

       Step 1: 定義 vocabulary size

       Step 2: 將 word 切成 character

       Step 3: 建立 language model

       Step 4: 選擇具有 maximum likelihood 的 subword

       Step 5: 重複 step 4，直到抵達 threshold

2. Answer Span :
    1. How did you convert the answer span start/end position on characters to position on tokens after BERT tokenization?
       我使用 example script 進行 BERT tokenization。在 example code 中,tokenizer 會返回一個"offset_mapping" 對應 token 和 character 的位置,用此方式來算出 BERT tokenization 後的 start/end position。
    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?

       我使用 example script 進行 predict。example code 會先以 start_logit 和 end_logit 進行相加算出 score。再以這個使用 LogSumExp trick 進行 normalize 算出 probability,取出 grid search 中 probability 最高者作為 final start/end position。

Q2: Modeling with BERTs and their variants

1. Describe
    1. your model (configuration of the transformer model)
       使用 bert-base-chinese model

```
{
  "_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": "4.17.0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
}
```

```
{
  "_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": "4.17.0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
}
```

    2. performance of your model
       Context selection (evaluation accuracy): 0.9491525292396545
       Question answering(evaluation):
           EM:0.7886340977068794
           F1:0.7886340977068794
           Kaggle Public Score:0.76220

    3. the loss function you used.
       Cross entropy loss

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

   optimization algorithm: Adam(lr=3e-5)

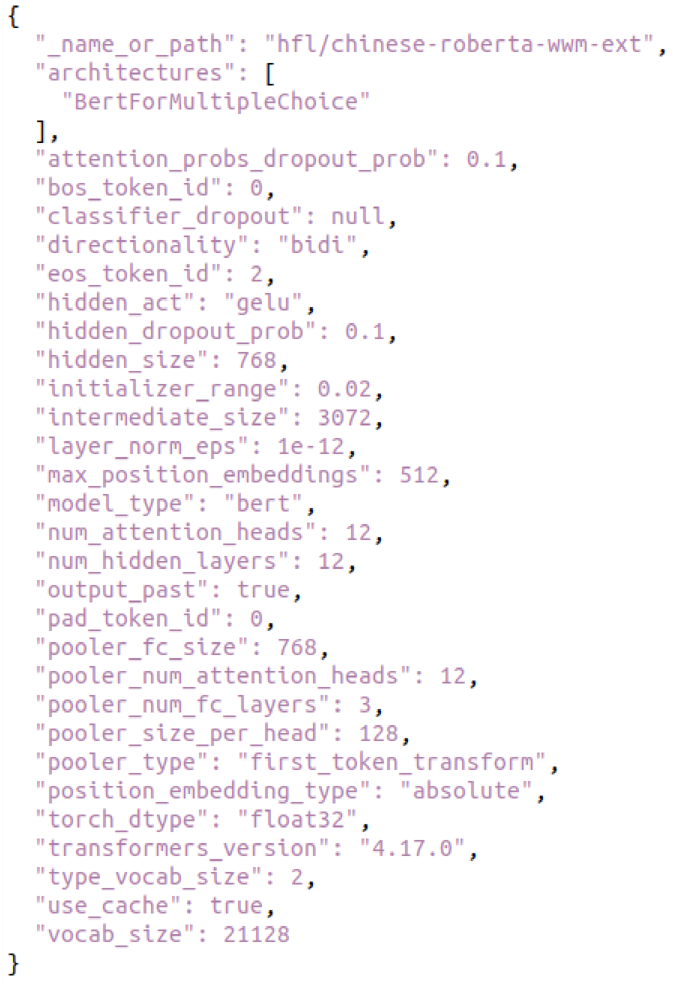
   lr scheduler: linear scheduler with warmup, warmup_ratio = 0.1

   batch size: 1

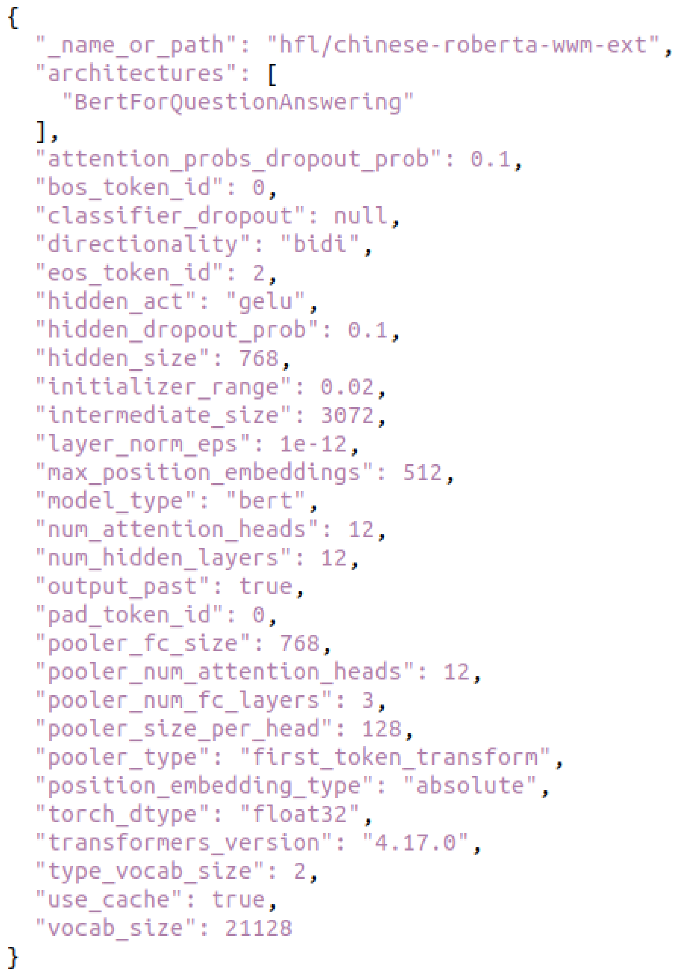
   gradient accumulation step: 2

2. Try another type of pretrained model and describe (2%)
   1. your model

      使用 hfl/chinese-roberta-wwm-ext



```
{
  "_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": "4.17.0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
}
```



```
{
  "_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": "4.17.0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
}
```

   2. performance of your model

      Context selection (evaluation accuracy): 0.9564639329910278

      Question answering(evaluation):

         EM:0.817215021601861

         F1:0.817215021601861

         Kaggle Public Score:0.78752

3. the difference between pretrained model (architecture, pretraining loss, etc.)

   以下為 RoBERTa 比 BERT 好的地方:

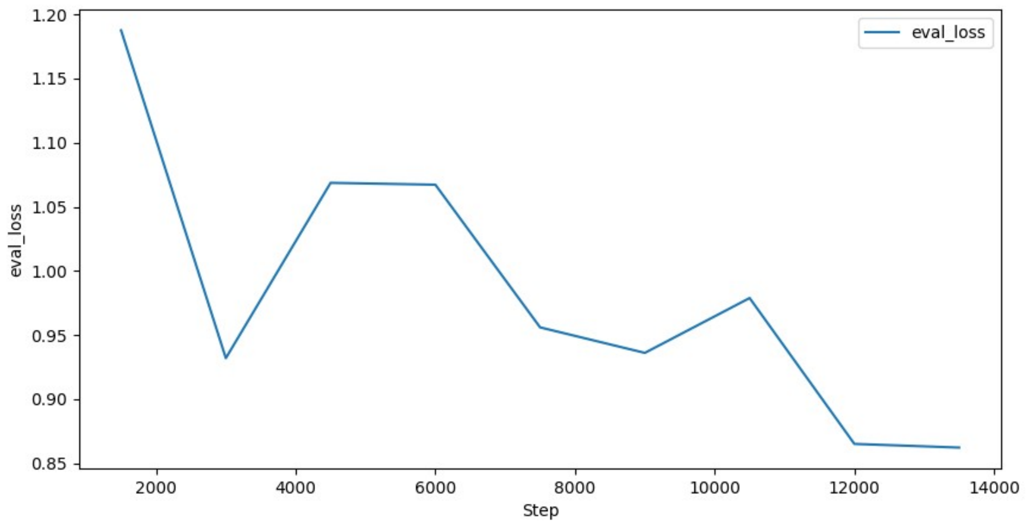
   1. 更多 training data(16G vs 160G)
   2. 使用 dynamic masking pattern 而不是 static masking pattern
   3. 把訓練目標從 next sentence prediction 換成 full sentences without NSP
   4. 在更長的 Sequences 上進行訓練

Q3: Curves

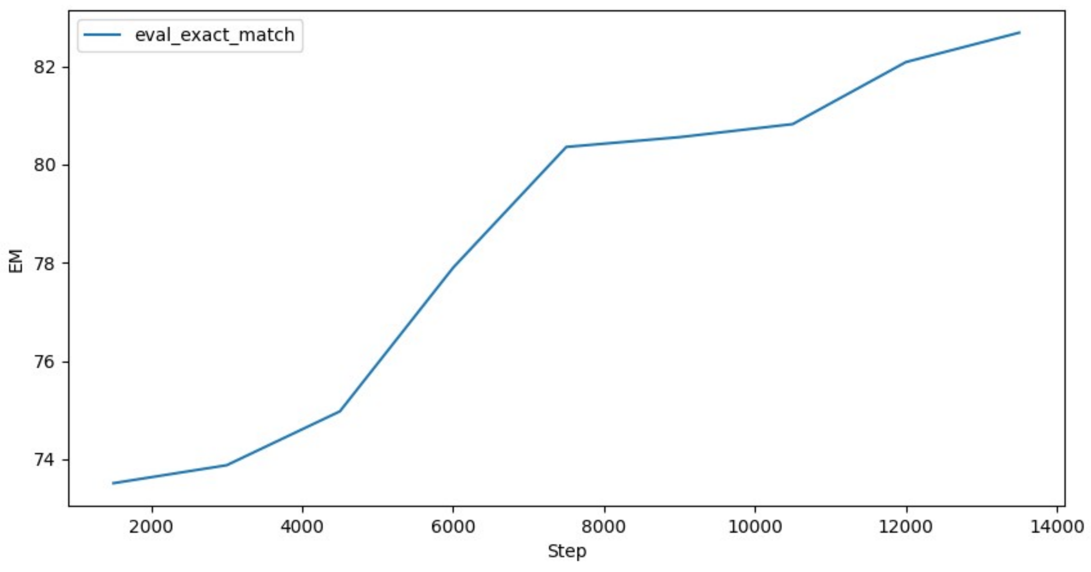
Plot the learning curve of your QA model

以下為 fine tune hfl/chinese-roberta-wwm-ext model 之 learning curve:

1. Learning curve of loss

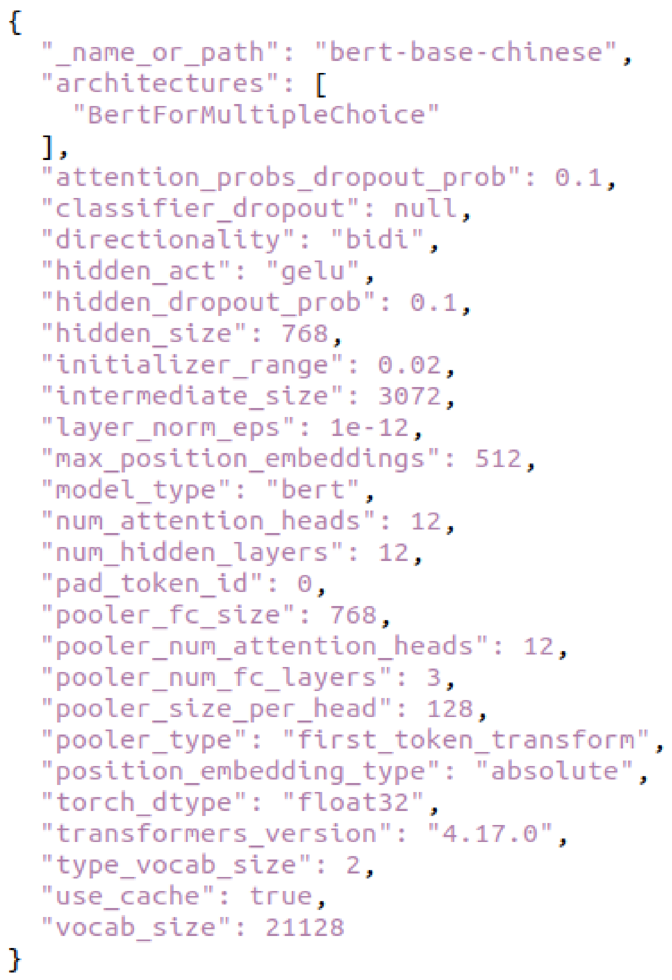


2. Learning curve of

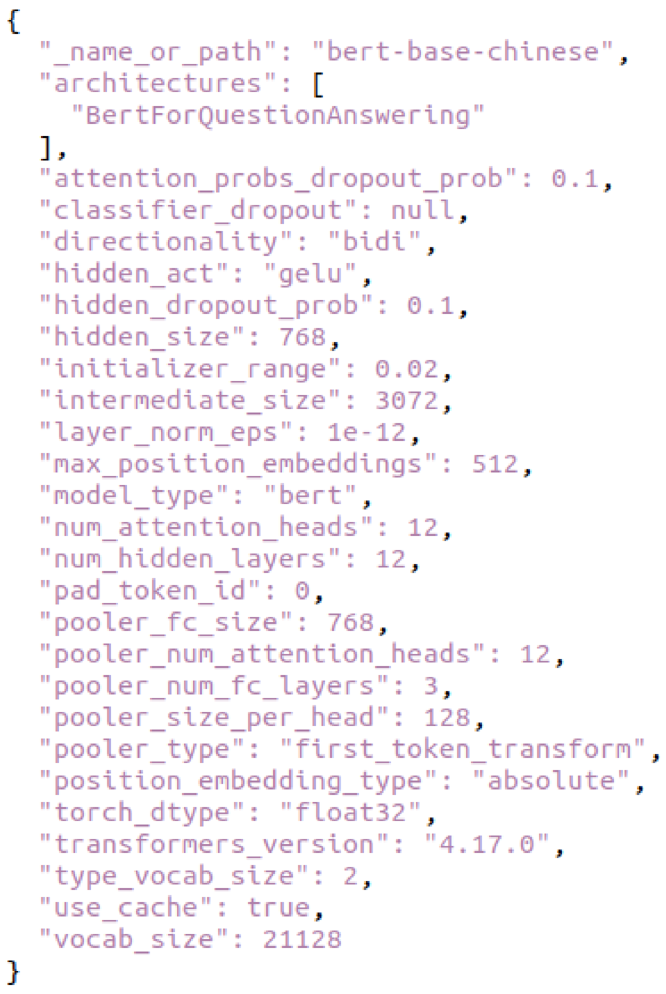


Q4: Pretrained vs Not Pretrained

1. model trained from scratch(configuration of the transformer model)
    1. configuration



```
{
  "_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": "4.17.0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
}
```



```
{
  "_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": "4.17.0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
}
```

    2. how to train this model
       我選擇以 Multiple choice task 進行 train from scratch。將 example script 中將
       model = AutoModelForMultipleChoice.from_pretrained 改成
       model=AutoModelForMultipleChoice.from_config(config)
2. performance comparison

	BERT	BERT(train from scratch)
Context selection(evaluation accuracy)	0.95	0.54
EM(evaluation)	0.79	0.79
F1(evaluation)	0.79	0.79
Kaggle Public Score:	0.76	0.43