Report

1-1.

Berttokenizer 分為 2 個部分，為 BasicTokenizer 和 WordpieceTokenizer。前者可以進行一些常規的操作，像是根據標點符號、空格進行分詞、unicode 轉換等等。後者就是把句子分成 1 個 1 個的單字。中文的話就是直接分成 1 個 1 個字。沒找到的字標記為未知，英文會根據分詞算法把一個單字分開，不是開頭的部分加上##。WordpieceTokenizer 可以對很多語言進行轉換。這次作業主要是對中文和數字進行切割。WordpieceTokenizer 的分詞算法就是老師上課所講的那樣，首先用 unicode 字符建立列表，根據 datasets 去看那些字符最常出現在一起，組合起來加字符表，最後達到數量上限為止。

1-2.

(a)

Qa 的部分，我是使用 sample code。Sample 中有用個東西叫 return_offsets_mapping，這個東西可以讓我們在比對 char start、char end 後找到答案在 input 中的位置， 然後判斷答案是否在 inputs 裡面。如果不在，就讓答案標註在 CLS 的位置。

(b)

做法是把 model 出來的 start_logits 和 end_logits 相加並記錄分

數，並刪去不合條件的例如 end<start 之類的，並用 n_best_size 紀錄比較好的結果，避免還原中分數最高結果的有錯，就可以還原下一個，依序下來就可以得到結果。

2.

(a)

我是用 chinese-roberta-wwm-ext 這個 model 過 strong baseline 的。configuration 如下:

這個 model 在 context_selection 的正確率是 0.951，在 qa 的正確率

是 0.8209。context_selection 的部分，直接看 BertForMultipleChoice 的源碼可知 loss function 是 CrossEntropyLoss。我用的 optimizer 是 Adam。Learning rate 是 3e-5，batch_size 是 1，不過我的 gradient accumulation step 是 4，所以 batch_size 綜合起來是 4。在 qa 的部分，直接看 BertForQuestionAnswering 的源碼可知 loss function 是 CrossEntropyLoss。optimizer 是 AdamW。Learning rate 是 3e-5，batch_size 是 4，gradient accumulation step 是 2，所以 batch_size 綜合起來是 8。

(b)

最基本的 model 是 bert-base-chinese。configuration 如下:

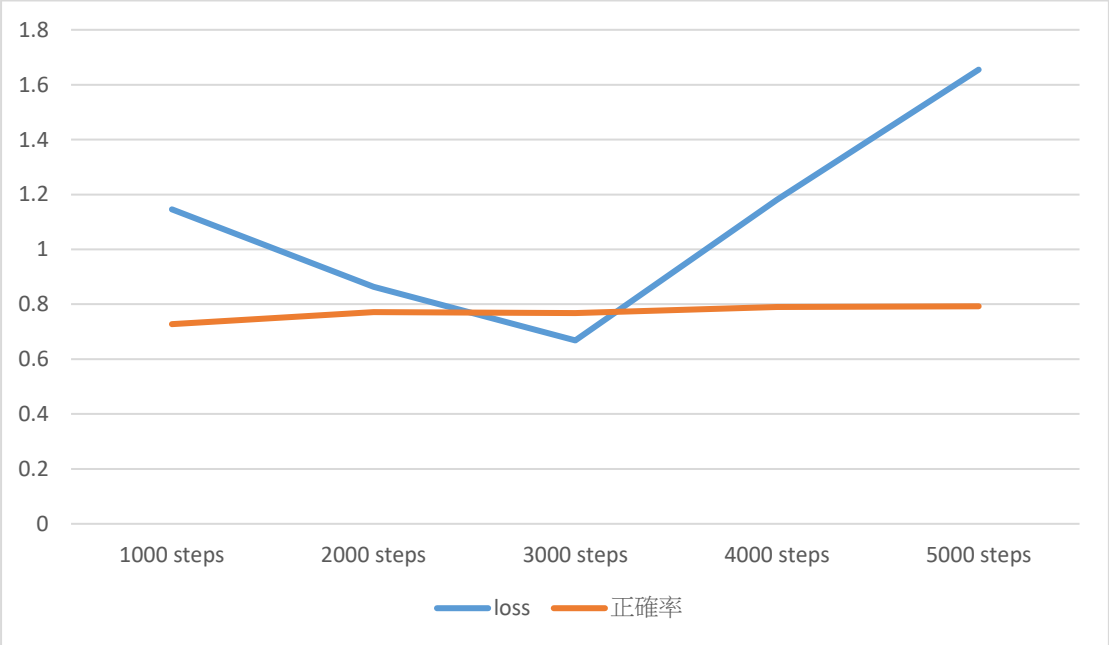

```
{
  "_name_or_path": "bert-base-chinese",
  "architectures": [
    "BertForMultipleChoice"
  ],
  "attention_probs_dropout_prob": 0.1,
  "classifier_dropout": null,
  "directionality": "bidi",
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "initializer_range": 0.02,
  "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": "4.22.2",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
}
```

這個 model 在 context_selection 的正確率是 0.946，在 qa 的正確率是 0.7912。

Roberta 有個特色是 dynamic masking。一般的 bert 訓練過程中不會改 mask 的位置。而 roberta 會進行動態調整。且 roberta 在訓練參數時使用了更多的資料。wwm 是指 Whole Word Masking，就是說原本 masked 是隨機的，所以單字在透過 Wordpiece 後可能一部分沒被 masked 一部份有。而 wwm 的設定就是如果一個單字有一部份被 masked，則整個單字會被 masked。

3.

曲線圖如下:



我合理懷疑 4000、5000 時遇到了奇怪的訓練資料，所以 loss 會上升

4.

我是在 MC 上跑的。單純把 BertForMultipleChoice.from_pretrained 換成

config = AutoConfig.from_pretrained(args.model_name)

model = AutoModelForMultipleChoice.from_config(config)就好了。其他參數都沒有動。Configuration 如下:


```
{
  "_name_or_path": "./roberta跑select",
  "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": "4.22.2",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
```


那個 path 是 chinese-roberta-wwm-ext 的 config

在 MC 跑 valid 的正確率是 0.516，感覺跟預訓練的模型相差很大。因為中文資料數量非常大，用 2 萬筆資料跑 2 小時很難跑出別人用上億筆資料跑很幾天甚至好幾個禮拜的結果，所以不容易有很好的結果。