ADL-HW2
B08305056，劉羽芯

1. Data preprocess

   A. Tokenizer

      i. 我們的原始輸入包含 4 個欄位，展示如下，我們的模型會把 context, question 兩個欄位的文本視為一個 pair 進行 tokenize

- 'id': ['593f14f960d971e294af884f0194b3a7'],
- 'context': ['在 19 世紀雙星觀測所獲得的成就.....測量出恆星的直徑。']
- 'question': ['舍本和誰的數據能推算出連星的恆星的質量？'],
- 'answers': {'answer_start': [108], 'text': ['斯特魯維']}

傳入文本以後 Tokenizer 會返回一個儲存了以下資訊的字典

- input_ids
- token_type_ids
- attention_mask
- offset_mapping
- overflow_to_sample_mapping

我們設定pad to left，所以question欄位會是第一個句子，context接在後面，並且context會根據設定的max_len, stride長度進行切片，，讓文本長度不會超過max_len，並且文本之間會有stride個字重疊。切片後的文本長度都會一樣，不足的補上[PAD]以下方為例：

切片: 0
長度: 200
[CLS] 舍 本 和 誰 的 數 據 能 推 算 出 連 星 的 恆 星 的 質 量 ？ [SEP] 在 19 世 紀 雙 星 觀 測 所..... 譜 線 時 發 現 第 一 顆 光 譜 雙 星 …… 第 一 個 獲 得 解 答 的 是 1827 年 由 菲 利 克 斯 · 薩 瓦 里 透 過 [SEP]

切片: 1

長度: 200
[CLS] 舍 本 和 誰 的 數 據 能 推 算 出 連 星 的 恆 星 的 質 量 ？ [SEP] 譜 線 時 發 現 第 一 顆 光 譜 雙 星 ，...... 第 一 個 獲 得 解 答 的 是 1827 年 由 菲 利 克 斯 · 薩 瓦 里 ......卡 爾 · 史 瓦 西 發 現 經 由 比 較 視 星 等 和 攝 影 星 等 的 差 別 ， 可 以 得 [SEP]

切片: 2
長度: 200
[CLS] 舍 本 和 誰 的 數 據 能 推 算 出 連 星 的 恆 星 的 質 量 ？ [SEP] 第 一 個 獲 得 解 答 的 是 1827 年 由 菲 利 克 斯 · 薩 瓦 里 …… 阿 爾 伯 特 · 邁 克 生 在 虎 克 望 遠 鏡 第 一 次 使 用 干 涉 儀 測 量 出 恆 星 的 直 徑 。 [SEP] [PAD] [PAD]

input_ids儲存的是每個字對應的id， offset_mapping則是儲存在tokenize以後，該字起始以及結束的位置，範例如下：

原始文本
[CLS] 舍 本 和 誰
input_ids
[101, 5650, 3315, 1469, 6306]
offset_mapping
[(0, 0), (0, 1), (1, 2), (2, 3), (3, 4)]

token_type_ids則會儲存哪些位置是question(1)，哪些位置是context(1)，spacial [SEP], [PAD](None)，延續上例，token_type_ids如下：

[None, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, None, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, None]

attention_mask則是儲存該位置是否需要被模型注意，如[PAD]、[CLS]等位置就會被忽略。設為0，其餘為1

B. answer span

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

在原始的輸入當中，我們可以知道答案在 context 當中開始的位置。而再透過測量答案的字串長度，我們可以得知答案在 context 當中結束的位置。

```
start_char = answers["start"][0]
end_char = start_char + len(answers["text"][0])
```

而 tokenize 以後的位置，則是透過 token_type_ids 獲得。
tokenize 以後的 context 開始位置，是在第一個 token_type_ids == 1 的位置，我們稱其為 token_start_idx，接著我們再根據原始 context 的長度得知 context tokenize 以後結束的位置，我們稱其為 token_end_idx。
接著我們透過 offset_mapping 的對應關係，確認原始答案在不在當前切片的文本範圍當中

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

我們設定了一個 n_best_size 的參數，挑出前 n_best_size 個可能性最高的答案開始跟結束 logit，並且將他們從小到大排序。

接著我們檢查每一種可能的開始、結束位置的組合，首先剔除掉那些結束比開始更早的答案，接著替每一種組合計算一個開始 logit probability + 結束 logit probability 的分數，最終挑選分數最高的組合

2. Modeling with BERTs and their variants

A. Describe

i. your model (configuration of the transformer model)

Multiple choice

Model config


```
{
   "architectures": [
      "BertForMaskedLM"
   ],
   "attention_probs_dropout_prob": 0.1,
   "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",
   "type_vocab_size": 2,
   "vocab_size": 21128}
```


Optimiser = ADAMW
Learning rate = 3e-05
per_device_train_batch_size = 1
per_device_eval_batch_size = 1
Loss function = CrossEntropyLoss

Performance = {'accuracy': 0.959122632103689}

Question answering

```
BertConfig {
  "_name_or_path": "hfl/chinese-roberta-wwm-ext-large",
  "architectures": [
    "BertForMaskedLM"
  ],
  "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",
  "transformers_version": "4.24.0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
}
```


performance of your model: 0.77938

CrossEntropyLoss

AdamW

Batch size = 8
Learning rate = 3e-05
per_device_train_batch_size = 8
per_device_eval_batch_size = 8

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

比較 bert base Chinese 以及 Bert base 在 QA 的表現

1. your model

The config of Bert base Chinese


```
{
  "architectures": [
    "BertForMaskedLM"
  ],
  "attention_probs_dropout_prob": 0.1,
  "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",
  "type_vocab_size": 2,
  "vocab_size": 21128
}
```


Performance: 0.7414

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

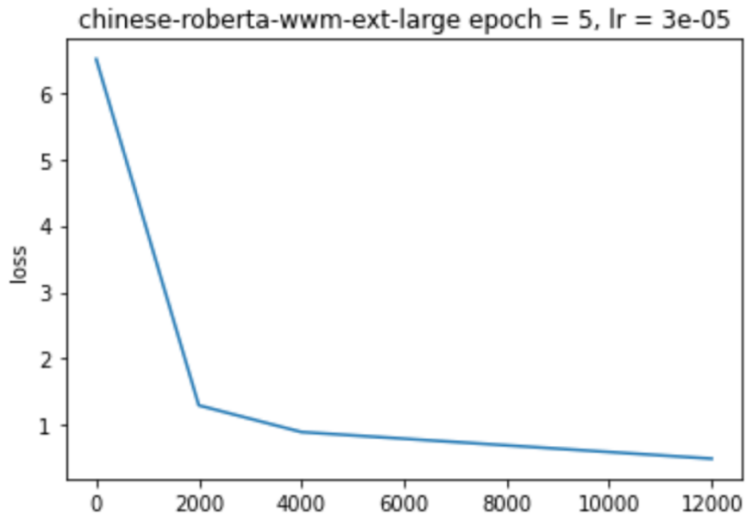
RoBERTa v.s BERT

Mask的方式是RoBERTa跟BERT最主要的差別，BERT是在預處理的時候一次進行mask，例如他會把單句話複製4次，然後在每個epoch訓練的時候重複使用這些副本，換句話說，BERT是採用靜態的方式進行mask。但是RoBERTa的mask是動態的，會在訓練當中進行，每一個epoch model可能可以看到mask了不同位置的文本，而且文本複製的次數也是不固定的。

BERT	RoBERTa
Static masking/substitution	Dynamic masking/substitution
Inputs are two concatenated document segments	Inputs are sentence sequences that may span document boundaries
Next Sentence Prediction (NSP)	No NSP
Training batches of 256 examples	Training batches of 2,000 examples
Word-piece tokenization	Character-level byte-pair encoding
Pretraining on BooksCorpus and English Wikipedia	Pretraining on BooksCorpus, CC-News, OpenWebText, and Stories
Train for 1M steps	Train for up to 500K steps

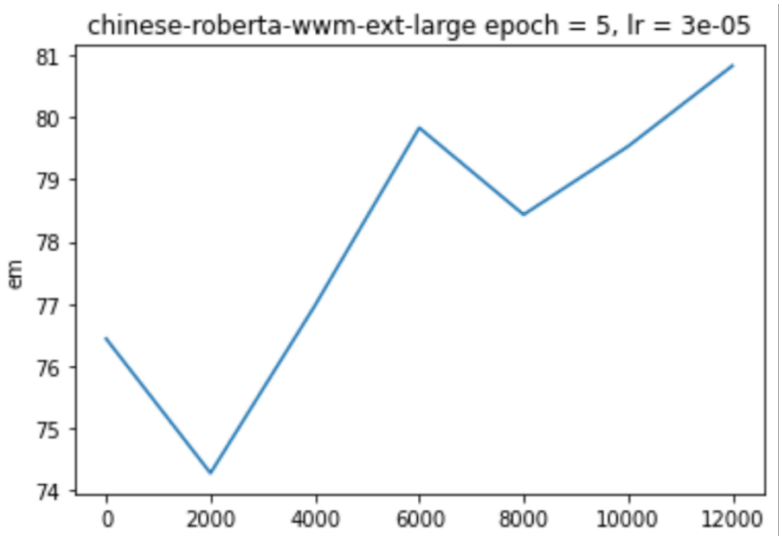
另外我們的 Roberta 模型是採用 wwm（whole word mask），也就是如果一個詞彙當中的子詞被遮蔽，則該詞的其他部分也會被遮蔽，以下比較 wwm 以及 bert tokenizer 的 mask 方式：
语 言 [M] 型 来 [M] 测 下 一 个 词 的 概 率 。
语 言 [M] [M] 来 [M] [M] 下 一 个 词 的 概 率 。
在 wwm 當中，「模型」這個詞彙下面，「模」跟「型」不會被拆開，會同時一起被遮蔽起來。

3. Curve
    A. Learning curve of loss (0.5%)



x軸為第x個batch

B. Learning curve of EM (0.5%)



x 軸為第 x 個 batch

4. Pretrained vs Not pretrain

我們使用前述的 bert base Chinese model

Number of epoch = 2
Learning rate = 3e-5
Max_len: 512
Batch_size: 2 (per_gpu_train_batch_size 1 * gradient_accumulation_steps 2)
Learning_rate: 3e-5

No pretrain

Model config BertConfig {
  "_name_or_path": "bert-base-chinese",
  "architectures": [
    "BertForMaskedLM"
  ],
  "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",
  "transformers_version": "4.24.0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
}

the performance of this model v.s. BERT

沒有 pretrain 的 model 在 kaggle 上的準確度是 0.06781，驗證資料集表現如下

{
"eval_exact_match": 5.284147557328016,
"eval_f1": 5.284147557328016
}

有經過 pretrain 的 bert 在 kaggle 上的準確度是 0.74141，驗證資料集表現如下

{
"eval_exact_match": 76.23795280824194,
"eval_f1": 76.23795280824194
}

5. Bonus: HW1 with BERTs

Slot tagging

A. Your model

Config

Model config BertConfig {
"_name_or_path": "bert-base-uncased",
"architectures": [
"BertForMaskedLM"
],
"attention_probs_dropout_prob": 0.1,
"classifier_dropout": null,
"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,
"position_embedding_type": "absolute",
"transformers_version": "4.24.0",
"type_vocab_size": 2,
"use_cache": true,
"vocab_size": 30522
}
Lr = 5e-5
per_device_train_batch_size = 8
per_device_eval_batch_size = 8

B. Performerance

C. Loss function

cross entropy loss

D. Optimize algorithm

AdamW

Intent

Config

Model config BertConfig {
"_name_or_path": "bert-base-uncased",
"architectures": [
"BertForMaskedLM"
],
"attention_probs_dropout_prob": 0.1,
"classifier_dropout": null,
"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,
    "position_embedding_type": "absolute",
    "transformers_version": "4.24.0",
    "type_vocab_size": 2,
    "use_cache": true,
    "vocab_size": 30522
  }
```


Lr = 2e-5

Optimizer = AdamW

Loss function = Cross Entropy Loss

Batch size = 128

Performance