ADL HW2

tags: `write-up`

Q1: Data processing (2%)

> model in Q1: hfl/chinese-lert-base

Tokenizer (1%)

首先先對 data 做 preprocessing，取出其文章編號 & 題目內容 & 文章內容。接者使用 AutoTokenizer 與預設 arguments，使文字能夠轉換成 tokenize。

tokenizer arguments

1. truncation strategy LongestFirst：以最長的字串優先。
2. truncation max_length 與 padding 都設定為 512；direction 也都為向右方 (Right)。
3. model & decoder 都使用 WordPiece，且對於 continuing subword 會以 ## 做額外標記。

補充: WordPiece

WordPiece 如同字面意思，就是將文字轉換成片狀，不過在中文、日文、韓文中，則是以 character-tokenized 轉換成單一個 character。此外，WordPiece 使得模型能夠分解目標文字為已知的子詞，以用來處理 tokenizer 以前從未見過的詞。總而言之一個字轉為一個 token，且會額外標註使否為單字 or 字詞。

tokenization algorithm

1. 設定 Tokenizer 規則，長度、strategy 等等。
2. 將字串切割成 character；eg. 字串中的 "我"，可能被轉換成 "我" or "##我"。
3. 在 model 中檢查可以轉換 character 至哪個 token。
4. 最後 padding & truncation by length & strategy。

tokenization algorithm input & output

> input

1. 文字字串：中文、數字、標點、符號。

> output

1. input_ids：目前 token 的編號。
    - [CLS] token_id ... token_id [SEP] token_id ... token_id [SEP] [PAD] ... [PAD]

2. token_type_ids：目前 token 為何種 type，表示目前是 Sequence A or B；換句話說，說明目前字串是否有換行。
3. attention_mask：目前 token 使用的 mask，表是否要參考目前的 token 與 padding 有關。

Answer Span (1%)

> 名詞解釋

offsets_mapping: a dict (char_start, char_end) for each token

a. convert the answer span start/end position on characters to position on tokens

1. 如果 return_offsets_mapping=True，則對於所有 offset_mapping 可以得知目前字串的解答，接者會做下列檢查。
2. 取出其對應的 answers，如果沒有 answers 則 start/end position of token 則給目前字串的 CLS token index。
3. 如果找到 answers 取出其 start_char，並以 answers 長度計算出 end_char。
4. 如果 start_char & end_char out of the span，則 start/end position of token 一樣給目前字串的 CLS token index。
5. 如果通過上述檢查，則迭代計算出 start/end position of token，如此可以將 position 限制在 current span 之中且不超出字串長度。

b. answer span start/end position to final start/end position

1. 由於一個題目可能不只一個預測結果 (n_pred)，下列說明處理對於所有預測結果的處理方法。
2. 如果此預測結果通過檢查則繼續計算機率大小，檢查：在字串範圍內，且長度不為負數，另外答案長度不超過 max_answer_length。
3. 比較各個預測結果的 logits 數值，將其透過 softmax 轉換成機率後，取有最大的機率的 start_position & end_position 作為答案。
4. 最後需要再透過 offset_mapping 轉換回 token / 文字串。

Q2: Modeling with BERTs and their variants (4%)

1. Describe (2%)

> model in Q2.1: ckiplab/bert-base-chinese

a. model configuration

> Multiple Choice

```
{
  "_name_or_path": "ckiplab/bert-base-chinese",
  "architectures": [
    "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
}
```

Question Answering

```
{
  "_name_or_path": "ckiplab/bert-base-chinese",
  "architectures": [
    "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
}
```

b. model performance

eval stage

type	acc / EM	loss
mc	0.9551	0.2412
qa	0.7866	1.0536

kaggle score

	Score
Public	0.76582
Private	0.76151

c. loss function

因為 `self.config.problem_type = "single_label_classification"` ，所以使用 torch.nn.CrossEntropyLoss。

d. The optimization algorithm, learning rate and batch size

> Multiple Choice

optimization algorithm: `torch.optim.AdamW(lr=3e-5)` lr scheduler: linear scheduler without warmup batch size: 1 gradient accumulation steps: 2

> Question Answering

same setting as Multiple Choice

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

> model in Q2.2: hfl/chinese-lert-base

a. model configuration

> Multiple Choice

```
{
  "_name_or_path": "hfl/chinese-lert-base",
  "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
}
```

> Question Answering

```
{
  "_name_or_path": "hfl/chinese-lert-base",
  "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": "4.22.2",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
}
```

b. model performance

eval stage

type	acc / EM	loss
mc	0.9670	0.1036
qa	0.8318	0.6077

kaggle score

	Score
Public	0.80289
Private	0.80849

c. the difference between pretrained model

architecture

> 由於 相關論文 22/11/11 才會於 arXiv 公布，所以此小題可能無法回答非常的完整與精確。
> Ref. GitHub - ymcui/LERT: LERT: A Linguistically-motivated Pre-trained Language Model

> by 作者 github

使用了下列方法，給予 model 更豐富的語言特徵：1. 語言學信息增強、2. MLM (Masked Language Model)、3. 3 種語言學任務、4. 語言學啟發的預訓練機制 (LIP)。

> by Me 推測

基於 BERT 模型之上，使用了對於特定詞性的 MASK (還是要看目前模型的任務是甚麼)，此外模型會參考 MLM (Masked Language Model)、POS (Part-of-Speech Tagging)、NER (Named Entity Recognition)、DEP (Dependencies) 等 features，給予了克漏字的能力、詞性標註與辨識能力、命名空間、更重要的是還有前後文的關係，如此使用更多語言特徵可以更準確的對語言任務有更好的表現。

train arguments

batch size: 4 gradient accumulation steps: 4 epochs in qa: 3

3. other models and performance

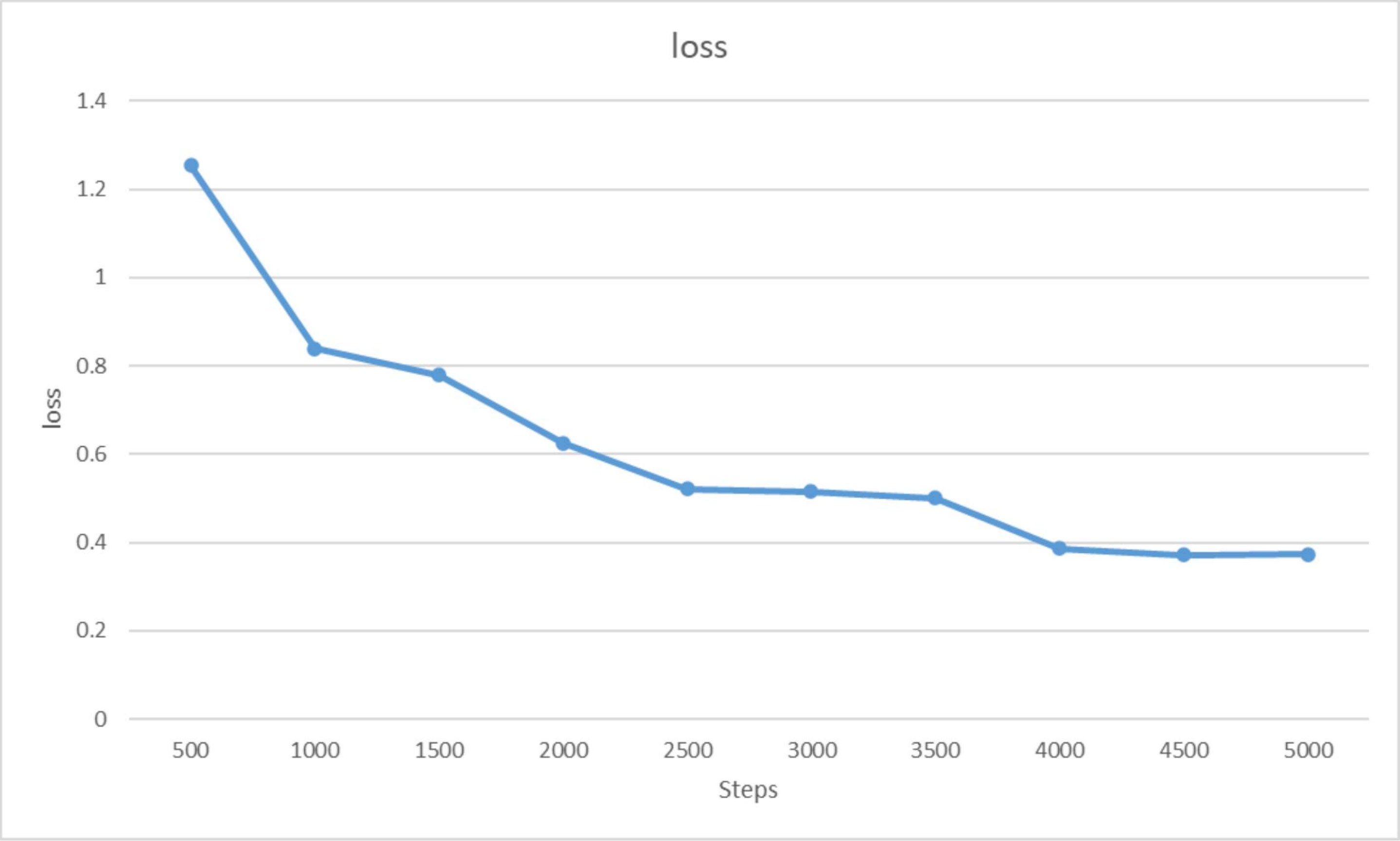
> 其他測試 model

name	Public	Private
ckiplab/bert-base-chinese	0.76582	0.76151
hfl/chinese-bert-wwm	0.75226	0.76422
hfl/chinese-bert-wwm-ext	0.75858	0.77235
hfl/chinese-roberta-wwm-ext	0.78119	0.79584
hfl/chinese-roberta-wwm-ext-large	0.79023	0.80487
hfl/chinese-macbert-base	0.78300	0.79855
hfl/chinese-macbert-large	0.78933	0.79674
hfl/chinese-lert-small	0.67811	0.67118
hfl/chinese-lert-base	0.80289	0.80849
hfl/chinese-lert-large	0.80379	0.81662
hfl/chinese-pert-base	0.77305	0.77868
hfl/chinese-pert-base-mrc	0.83815	0.85094
hfl/chinese-pert-large	0.26491	0.28455
hfl/chinese-pert-large-mrc	0.85443	0.85817
hfl/chinese-xlnet-mid	0.25316	0.26287

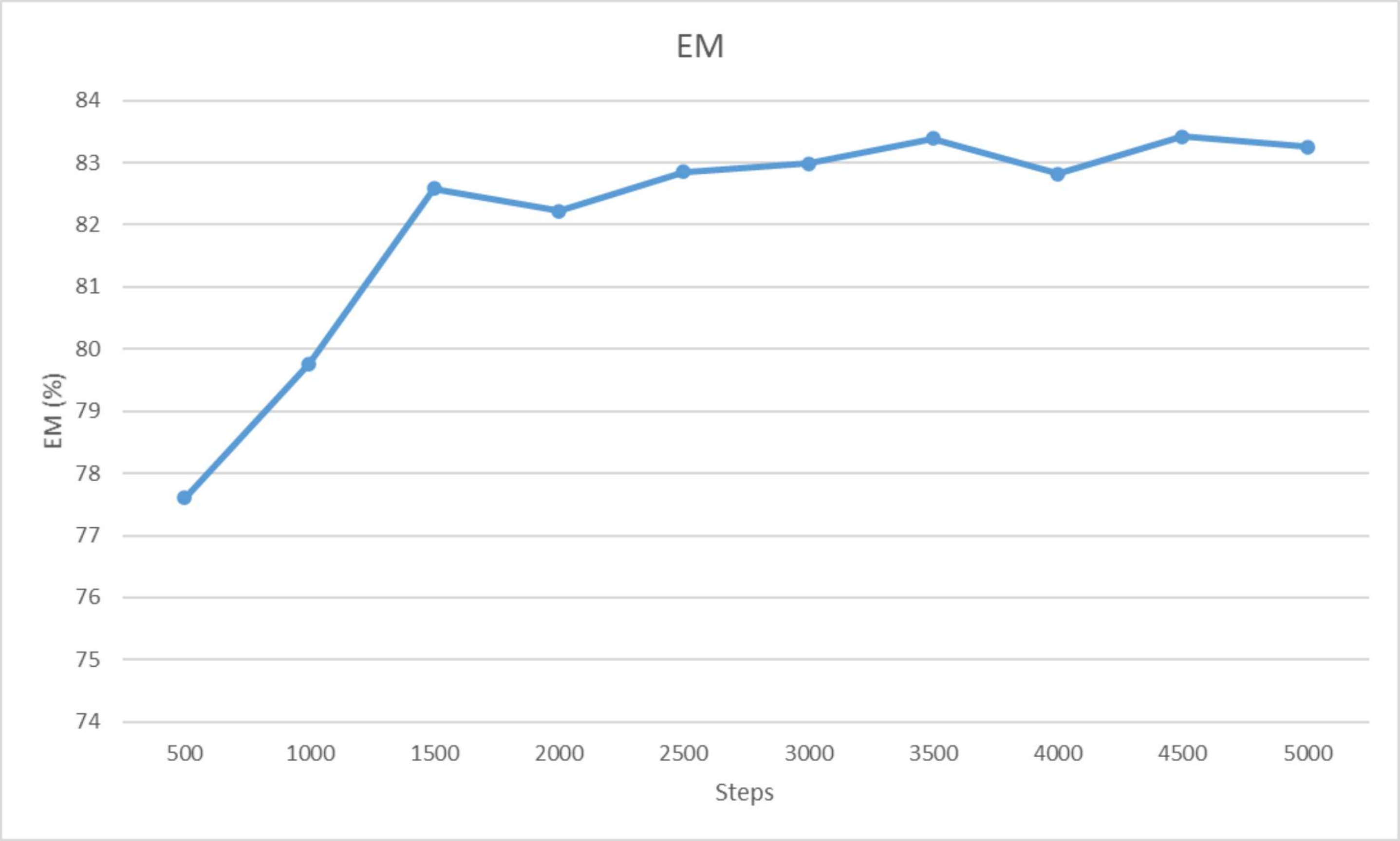
Q3: Curves (1%)

> QA model in Q3: hfl/chinese-lert-base

a. Learning curve of loss (0.5%)



b. Learning curve of EM (0.5%)



Q4: Pretrained vs Not Pretrained (2%)

> compare with: model in Q2.1 (bert)

a. model configuration

> Multiple Choice

```
{
  "architectures": [
    "BertForQuestionAnswering"
  ],
  "attention_probs_dropout_prob": 0.1,
  "classifier_dropout": null,
  "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",
  "torch_dtype": "float32",
  "transformers_version": "4.22.2",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 30522
}
```

> Question Answering

```
{
  "architectures": [
    "BertForMultipleChoice"
  ],
  "attention_probs_dropout_prob": 0.1,
  "classifier_dropout": null,
  "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",
    "torch_dtype": "float32",
    "transformers_version": "4.22.2",
    "type_vocab_size": 2,
    "use_cache": true,
    "vocab_size": 30522
  }
```

b. model performance compare w/ Q2.1 bert model

eval stage

type	acc / EM	loss
bert mc	0.9551	0.2412
bert qa	0.7866	1.0536
my bert mc	0.5357	0.9829
my bert qa	0.0468	4.6073

kaggle score

	Public Score	Private Score
bert	0.76582	0.76151
my bert	0.03616	0.02710

c. compare (小結)

訓練參數在與 bert 的訓練參數相同的情況下，雖然 train from scratch 一樣可以使得 loss 下降，使得 Multiple Choice acc. & Question Answering EM 可以上升，這是因為模型設計本來就很優秀，但是如果想要從這麼小的資料集，卻想要能夠訓練出完整 / 完美的語言模型想必是非常困難的。

不過其實受限於訓練時間與資源的關係，如果有機會可以增加訓練的 epoch 數量、改變 batch size & gradient accumulation steps，說不定有機會可以使得 kaggle 分數再度提高。