深度學習應用 - 作業二

tags: ADL

系級：電機所碩一　姓名：楊冠彥　學號：R11921091

Q1: Data processing

1. Tokenizer

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

我直接使用Hugging Face寫好的code來使用Bert的Tokenizer。 Tokenizer的目標是將 token 編碼成 ids，再經過 embedding layer 將 token 轉為帶有詞語意義的word vector給模型學習。

`bert-base-chinese` 及幾乎所有中文BERT模型的tokenizer都是character-based。也就是說，中文字元會被視作token。而這些Bert類型的模型使用的是WordPiece Tokenizer，Google至今未開源其 WordPiece 訓練算法的實作程式，但根據Hugging Face的推測，WordPiece Tokenizer會先定義 vocabulary大小以包含資料集中存在的每個字元，並將word分割成字元，再根據字元資料選擇要合併 pair的方式，並建立模型，重覆透過選擇能增加最大概似的 subword 直到達到threshold。

補：WordPiece 使用以下公式計算每pair的分數：

$$score = (freq - of - pair)/(freq - of - first - element \times freq - of - second - element)$$

這個算法會優先合併單個部分在volcabulary table中頻率較低的pair。
公式來源：Hugging Face WordPiece Course

2. Answer Span

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

在 tokenizer 設定 return_overflowing_tokens=True, return_offsets_mapping=True。 return_offsets_mapping 可以幫助我們從 character 轉到 token 的 start end。Tokenized data 會包含每個 token 對應 question 或 context 的 (char start position, char end position)，只要迭代找出 span start 與 char start 相同的位置便為 start position，span end 與 char end 相同的位置即為 end position。

return_overflowing_tokens 則是為了將 sequence mapping 回去，因為較長的 sequence 會被切成多份，這時就只要再額外做一些 start end position 的判斷就完成了。

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?

模型output時，需要判斷 start/end position 不合理的狀況，以及不同的 start/end position 組合，做法就是對每組 start/end 的配對機率相乘(沒有過 exponential的則為相加)。再將 start position 和 end position 的分數高到低進行排序後，找出兩兩配對最好的前幾名，並濾掉長度不合理或是 end position < start position 的狀況。

Q2: Modeling with BERTs and their variants

1. Describe

先嘗試基本的模型Hugging Face的 bert-base-chinese。

a. model

首先我們知道BERT 的input sequence會以 [CLS] (question) [SEP] (context) [SEP] [PAD]...[PAD]這樣的樣子構造，其中 [CLS] 、 [SEP] 和 [PAD] 是特殊的token，[PAD]將不足的部分補到最大sequence長度，[CLS]和[SEP]分別是tokenized question和context sequence。

`Context Selection`

簡單來說，Bert在輸入sequence這個部分會需要以下四項：
Token embedding sequence (Emb_T)：訓練這個 token 本身的含義
Segment embedding sequence (Emb_S)：訓練來分辨 token 是屬於哪個句子，所以只有兩種不同 embedding
Position embedding sequence (Emb_P)：訓練分辨不同 token 的位置
Attension mask sequence ($Mask_{Att}$)：將所有sequence填充到相同的長度，將 tokens_tensors 裡頭不為 zero padding的位置設為 1 讓 BERT 只注意這些位置的 tokens。

而根據老師10/13上課影片，可推論每個 $(Emb_T, Emb_S, Emb_P, Mask_{Att})$ 都要經過BERT encoder進行特徵提取。第一個token的最後一層隱藏狀態，即 [CLS] ，先經線性層處理，再透過一個 Tanh activation function，計算出intent類別的 logits。
以下為上面提到整個過程的式子：

$$h_{00}, h_{01}, \ldots, h_{N0} = Bert(Emb_{Tj}, Emb_{Sj}, Emb_{Pj}, Mask_{Attj})$$
$$y_j = Tan(Linear(h_{N0}))$$

其中，
h_{it}:表示sequence中第t個token的第j層隱藏狀態
N：模型層數
y_j：表示是每個問題與其第j個context之間的相關logit

- 架構
  config.json

```
{
  "_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
}
```


`Questoin Answering`

這裡和前面Multichoice有些小不同，每個問題只有一個相關的context，也就是訓練期間來自資料集的groundtruth，或者在inference期間由模型預測2的。每個context關係的$(Emb_T, Emb_S, Emb_P, Mask_{Att})$一樣都要經過BERT encoder進行特徵提取。接著，每個token的最後一層隱藏狀態由兩個不同的linear layer進一步處理，分別計算起始位置logit及計算答案範圍的結束位置logit。也就是開始/結束位置 logit 表示特定token是開始/結束位置答案範圍的可能性。
以下為上面提到整個過程的式子：

$$h_{00}, h_{01}, \ldots, h_{N0}, \ldots, h_{Nt}, \ldots, h_{NL-1} = Bert(Emb_{Tj}, Emb_{Sj}, Emb_{Pj}, Mask_{Attj})$$
$$starty_t = Tan(Linear(h_{Nt}))$$
$$endy_t = Tan(Linear(h_{Nt}))$$

其中，
h_{it}:表示sequence中第t個token的第j層隱藏狀態
N：模型層數
$starty_j$：token t的起始位置logit
$endy_t$：token t的結束位置logit
L：表最大sequence長度

- 架構
  config.json


```
{
  "_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": "4.22.2",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
}
```


b. Performance

- Context selection accuracy: 0.95314
- Question answering EM: 0.79561
- Question answering F1: 0.86168
- Public accuracy on Kaggle: 0.74231
- Private accuracy on Kaggle: 0.76242

c. Loss function

Context Selection 和 Questoin Answering我都使用標準Cross entropy loss
$Loss = CrossEntropyLoss(y*, gt)$， $y*$為 intent classifier output， gt 為 ground truth 。

d. Optimization algorithm, learning rate and batch size etc.

`Context Selection`

- Optimization algorithm: AdamW
- Learning rate: 3e-5 = 0.00003
- Batch_size: 2 (per_gpu_train_batch_size 1 * gradient_accumulation_steps 2)
- Num_train_epochs: 3
- Max_len: 512

`Questoin Answering`

- Optimization algorithm: AdamW
- Learning rate: 3e-5 = 0.00003
- Batch_size: 2 (per_gpu_train_batch_size 1 * gradient_accumulation_steps 2)
- Num_train_epochs: 1
- Max_len: 512

2. Try another type of pretrained model and describe

參考老師10/20的上課影片，我嘗試了Hugging Face的 hfl/chinese-roberta-wwm-ext 、hfl/chinese-roberta-wwm-ext-large 、hfl/chinese-macbert-base 、hfl/chinese-macbert-large 、 hfl/chinese-pert-base 、 hfl/chinese-lert-base 、hfl/chinese-lert-large 的模型，最後選擇在Context Selection使用hfl/chinese-macbert-large ；Question Answering使用hfl/chinese-lert-large ，如對前述我嘗試過模型的Performace有興趣，可查看本份文件最下方的 **Experiment Result** ，我將Performace和參數配置簡單整理在那裡。

a. model

`Context Selection`

每個 $(Emb_T, Emb_S, Emb_P, Mask_{Att})$ 都要經過BERT encoder進行特徵提取。第一個token的最後一層隱藏狀態，即 [CLS] ，先經線性層處理，再透過一個 Tanh activation function，計算出intent類別的logits。
以下為上面提到整個過程的式子：

$$h_{00}, h_{01}, \ldots, h_{N0} = Chinese - Macbert - large(Emb_{Tj}, Emb_{Sj}, Emb_{Pj}, Mask_{Attj})$$
$$y_j = Tan(Linear(h_{N0}))$$

其中，
h_{it}:表示sequence中第t個token的第j層隱藏狀態
N：模型層數
y_j：表示是每個問題與其第j個context之間的相關logit

- 架構

config.json

```
{
  "_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": "4.22.2",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
}
```


`Questoin Answering`

這裡我使用了Hugging Face的 hfl/chinese-lert-large ，這裡和Q2一樣，每個問題只有一個相關的context，也就是訓練期間來自資料集的groundtruth，或者在inference期間由模型預測2的。每個context關係的 $(Emb_T, Emb_S, Emb_P, Mask_{Att})$ 一樣都要經過BERT encoder進行特徵提取。接著，每個token的最後一層隱藏狀態由兩個不同的cosine layer進一步處理，分別計算起始位置logit及計算答案範圍的結束位置logit。也就是開始/結束位置 logit 表示特定token是開始/結束位置答案範圍的可能性。
以下為上面提到整個過程的式子：

$$h_{00}, h_{01}, \ldots, h_{N0}, \ldots, h_{Nt}, \ldots, h_{NL-1} = Chinese - Lert - Large(Emb_{Tj}, Emb_{Sj}, Emb_{Pj}, Mask_{Attj})$$

$$starty_t = Tan(Linear(h_{Nt}))$$
$$endy_t = Tan(Linear(h_{Nt}))$$

其中，
h_{it}:表示sequence中第t個token的第j層隱藏狀態
N：模型層數
$starty_j$：token t的起始位置logit
$endy_t$：token t的結束位置logit
L：表最大sequence長度

- 架構

config.json


```
{
  "_name_or_path": "hfl/chinese-lert-large",
  "architectures": [
    "BertForQuestionAnswering"
  ],
  "attention_probs_dropout_prob": 0.1,
  "classifier_dropout": null,
  "directionality": "bidi",
  "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,
  "pad_token_id": 0,
```

```
  "pooler_fc_size": 1024,
  "pooler_num_attention_heads": 16,
  "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. Performance

- Context selection accuracy: 0.96610
- Question answering EM: 0.84978
- Question answering F1: 0.91329
- Public accuracy on Kaggle: 0.80831
- Private accuracy on Kaggle: 0.81752

c. Loss function

Context Selection和Questoin Answering我都使用標準Cross entropy loss
$Loss = CrossEntropyLoss(y*, gt)$， $y*$為 intent classifier output， gt 為 ground truth 。

d. Optimization algorithm, learning rate and batch size etc.

Context Selection

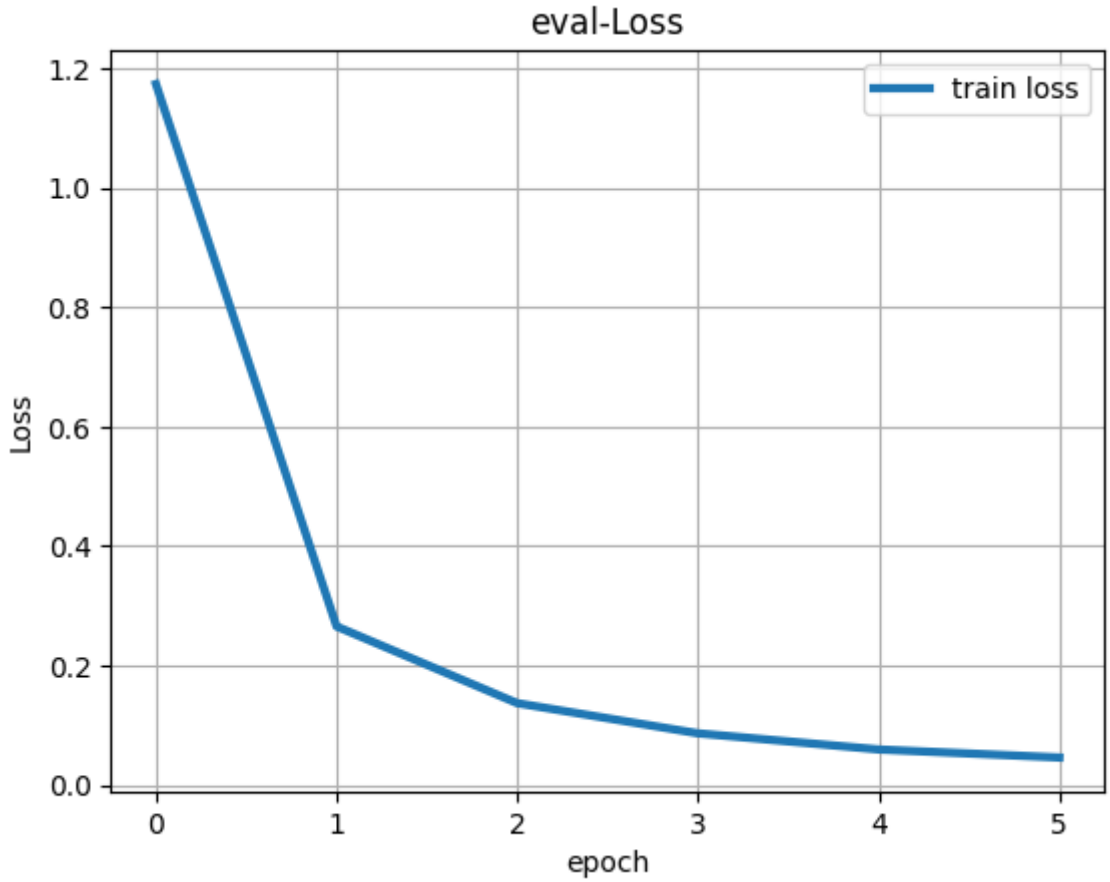
- Optimization algorithm: AdamW
- Learning rate: 3e-5 = 0.00003
- Batch_size: 8 (per_gpu_train_batch_size 1 * gradient_accumulation_steps 2)
- Num_train_epochs: 2
- Max_len: 512

Questoin Answering

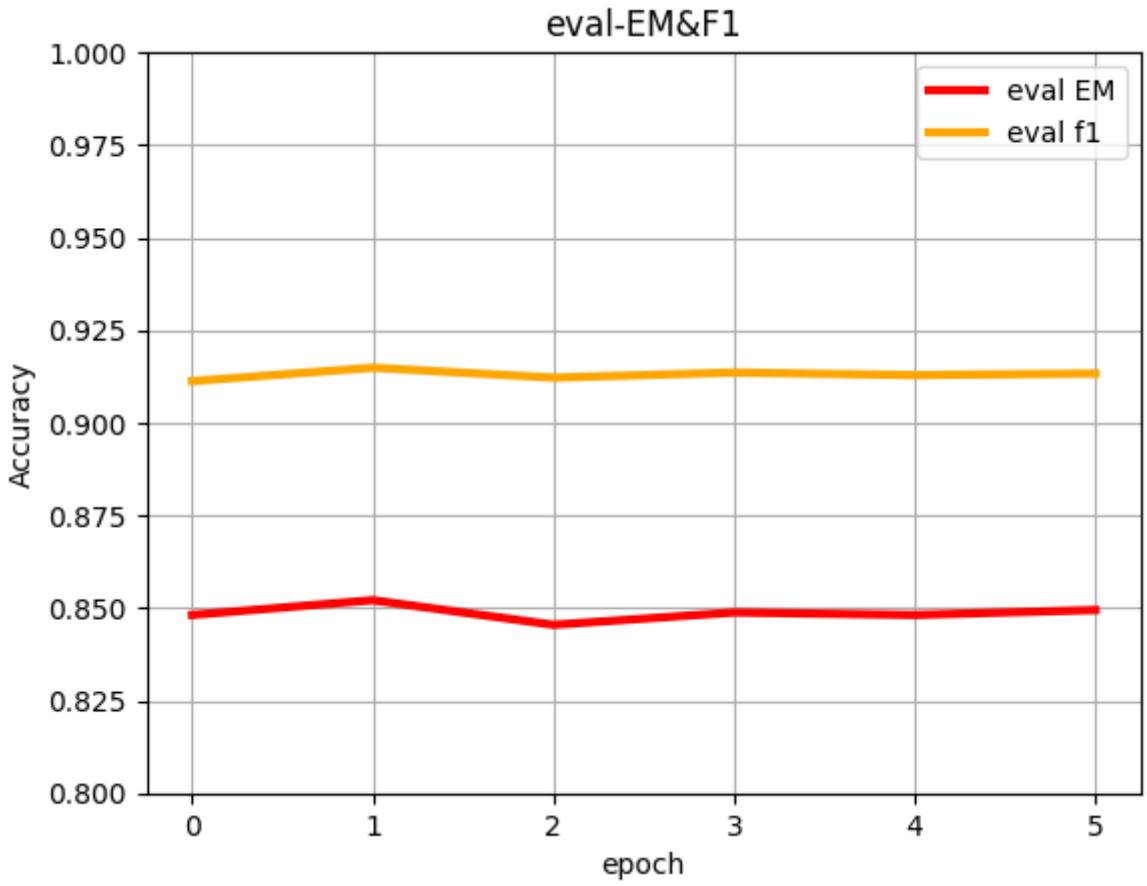
- Optimization algorithm: AdamW
- Learning rate: 3e-5 = 0.00003
- Batch_size: 64 (per_gpu_train_batch_size 1 * gradient_accumulation_steps 8)
- Num_train_epochs: 6
- Max_len: 512

Q3: Plot the learning curve of my QA model

a. Learning curve of loss



b. Learning curve of EM&F1



Q4: Pretrained vs Not Pretrained

這部分我將Quetion Answering部分跑Not Pretrained的狀況，並與Pretrained狀況比較。
其實就是AutoModelForQuestionAnswering.from_pretrained註解掉改成
AutoModelForQuestionAnswering.from_config，如下：

run_qa_no_trainer.py

```
    config = AutoConfig.from_pretrained(args.model_name_or_path)
    tokenizer = AutoTokenizer.from_pretrained(args.model_name_or_path,
use_fast=True)

    # Pretrained
    # model = AutoModelForQuestionAnswering.from_pretrained(
    #             args.model_name_or_path,
    #             from_tf=bool(".ckpt" in args.model_name_or_path),
    #             config=config,
    #         )

    # Not Pretrained
    model = AutoModelForQuestionAnswering.from_config(config)
```

a. Not Pretrained architecture

hidden_size: 768 → 32

num_attention_heads: 12 → 4

num_hidden_layers: 12 → 4

- 架構

config.json

```
{
  "_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": 32,
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 4,
  "num_hidden_layers": 4,
  "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. Loss function

與第2題描述的Loss function相同
Questoin Answering我都使用標準Cross entropy loss
$Loss = CrossEntropyLoss(y*, gt)$， $y*$為 intent classifier output， gt 為 ground truth 。

c. Optimization algorithm, learning rate and batch size etc.

與第2題描述的參數都相同

- Optimization algorithm: AdamW
- Learning rate: 3e-5 = 0.00003
- Batch_size: 2 (per_gpu_train_batch_size 1 * gradient_accumulation_steps 2)
- Num_train_epochs: 1
- Max_len: 512

d. Compare the performance with Pretrained

Qustion Answering	Context Selection	Public EM	Public F1
bert-base-chinese	bert-base-chinese	0.79561	0.86168
Not Pretrained	bert-base-chinese	0.03755	0.0707

明顯可以見到準確度差異非常大，即使我把Qustion Answering Not Prtrained的模型num_train_epochs再改成10，Public EM也僅到0.04320，這是由於 Trainsformer 架構龐大，需要大量訓練資料與時間才能訓練得起來，而本次訓練資料集相對非常小，因此容易較快出現overfitting的狀況，而且我們也沒有足夠算力或是時間來重頭訓練Bert模型。

Q5: Bonus: HW1 with BERTs

a. model

`Intent Classification`

這裡使用 bert-base-uncased pretrained model 整個過程與第二題的context selection差不多，可用以下式子描述：

$$h_{00}, h_{01}, \ldots, h_{N0} = Bert(Emb_{Tj}, Emb_{Sj}, Emb_{Pj}, Mask_{Attj})$$
$$y(0), \ldots, y(149) = Tan(Linear(h_{N0}))$$

`Slot tagging`

這裡使用 bert-base-uncased pretrained model 整個過程與第二題的question answering差不多，可用以下式子描述：

$$h_{00}, h_{01}, \ldots, h_{N0}, \ldots, h_{Nt}, \ldots, h_{NL-1} = Bert(Emb_{Tj}, Emb_{Sj}, Emb_{Pj}, Mask_{Attj})$$
$$y(t0), \ldots, y(t9) = Tan(Linear(h_{Nt}))$$

b. Loss function

Context Selection和Questoin Answering我都使用標準Cross entropy loss
$Loss = CrossEntropyLoss(y*, gt)$， $y*$為 intent classifier output， gt 為 ground truth 。

c. Optimization algorithm, learning rate and batch size etc.

Intent Classification

- Optimization algorithm: AdamW
- Learning rate: 1e-5 = 0.00001
- Batch_size: 32
- Num_train_epochs: 8
- Max_len: 128

Slot tagging

- Optimization algorithm: AdamW
- Learning rate: 1e-5 = 0.00001
- Batch_size: 32
- Num_train_epochs: 8
- Max_len: 128

d. Performace & Compare with my final model in HW1

`Intent Classification`

- HW1

Name	Number of Layer	Hidden Size	Best Epoch	Train Loss	Train Acc.	Val. Loss	Val. Acc.
LSTM	*2**	256	70	0.0502	0.9996	0.72006	0.92

- Comparing Table

Name	Train Loss	Train Acc.	Val. Loss	Val. Acc.	Public Acc. on Kaggle	Private Acc. on Kaggle
LSTM	0.0502	0.9996	0.72006	0.92	0.91511	0.91466
bert-base-uncased	0.19185	0.99667	0.24433	0.967333	0.96133	0.96088

`Slot tagging`

- HW1

Name	Number of Layer	Hidden Size	Best Epoch	Train Loss	Train Acc.	Val. Loss	Val. Acc.
GRU	*2**	512	100	0.002134	0.947404	0.017795	0.841

- Comparing Table

Name	Train Loss	Train Acc.	Val. Loss	Val. Acc.	Public Acc. on Kaggle	Private Acc. on Kaggle
GRU	0.002134	0.947404	0.017795	0.841	0.83217	0.81511
bert-base-uncased	0.00581	0.80094	0.00683	0.752	0.74852	0.7717

Experiment Result

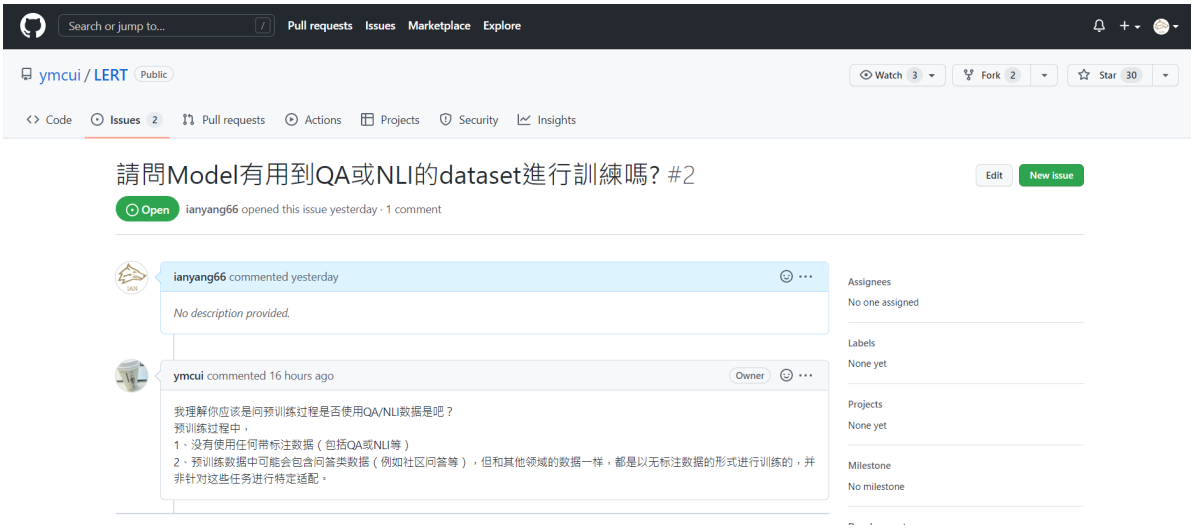
Model Name (CS)	Model Name (QA)	Gradient Accumulation Steps (MC)	Gradient Accumulation Steps (QA)	Num Train Epochs (CS)	Num Train Epochs (QA)	lr scheduler type (CS)	lr scheduler type (QA)	Public Acc. on Kaggle
bert-base-chinese	bert-base-chinese	2	2	3	1	linear	linear	0.74231
hfl/chinese-roberta-wwm-ext	hfl/chinese-roberta-wwm-ext	2	2	3	3	linear	linear	0.76401
hfl/chinese-roberta-wwm-ext	hfl/chinese-roberta-wwm-ext	2	2	3	3	linear	cosine	0.783
hfl/chinese-roberta-wwm-ext	hfl/chinese-roberta-wwm-ext	2	2	3	20	linear	cosine	0.77215
hfl/chinese-roberta-wwm-ext	hfl/chinese-roberta-wwm-ext	2	2	3	20	linear	linear	0.7613
hfl/chinese-roberta-wwm-ext	hfl/chinese-pert-base	2	8	3	10	linear	linear	0.76672
hfl/chinese-roberta-wwm-ext	hfl/chinese-lert-base	2	8	3	10	linear	cosine	0.79023
hfl/chinese-roberta-wwm-ext	hfl/chinese-lert-base	2	8	3	10	linear	linear	0.79023
hfl/chinese-roberta-wwm-ext	hfl/chinese-lert-base	2	8	3	15	linear	linear	0.79023
hfl/chinese-roberta-wwm-ext	hfl/chinese-lert-base	2	8	3	20	linear	linear	0.78119
hfl/chinese-roberta-wwm-ext	hfl/chinese-lert-base	2	8	5	10	linear	linear	0.78661
hfl/chinese-roberta-wwm-ext	hfl/chinese-lert-large	2	8	3	6	linear	linear	0.79475
hfl/chinese-roberta-wwm-ext-large	hfl/chinese-lert-large	8	8	2	6	linear	linear	0.79385
hfl/chinese-macbert-base	hfl/chinese-lert-large	8	8	6	6	linear	linear	0.79927
hfl/chinese-macbert-large	hfl/chinese-lert-large	8	8	2	6	linear	linear	0.80831
hfl/chinese-macbert-large	hfl/chinese-lert-large	8	64	2	10	linear	linear	0.80289

CS: Context Selection

QA: Question Answering

Appendix

針對lert模型已於github詢問哈工大訊飛聯合實驗室，確認非QA/NLI模型。



Reference

- LeeMeng - 進擊的 BERT：NLP 界的巨人之力與遷移學習
- Transformer github
- Hugging Face
- Revisiting Pre-Trained Models for {C}hinese Natural Language Processing
- HFL Github