網媒所碩一 R10944028 蔡易儒

ADL HW2 report

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.

因為相較於其他語言，中文帶有文意的基本單位是詞，因此使用傳統的中文字分割工具，先將句子分割成中文語意的單詞，而不是切成 word 或是 subword。使用的工具為 LTP 哈工大社會計算與信息檢所研究中心的語言技術平台（https://www.ltp-cloud.com/intro#ltp）。而 Bert Tokenizer 則是直接將一個中文字就當作一個 token 來轉換。

舉例來說：“使用语言模型来预测下一个词的概率。”

CWS　　：“使用 语言 模型 来 预测 下 一个 词 的 概率 。”

BERT Tokenizer：“使 用 语 言 模 型 来 预 测 下 一 个 词 的 概 率 。”

根據哈工大社會計算與信息檢所研究中心的網站介紹，他們的分詞演算法應該是使用傳統的中文分詞系統，然而其並沒有詳細的介紹，因此簡單的介紹我認為比較有可能被使用到的方法，其中滿有可能的是利用最大匹配法，從左到右每次取辭典中最長的字的長度，與辭典中的詞作比較若沒有符合則將長度減一，繼續搜索直到找到或是詞長度為一，其中也有分正向與反向，正向即是從左到右，而反向則是從右到左。通常會同時考慮正反方向，當兩演算法的結果不同時，通常會優先考慮分詞較少的方向，如句子為”我們在野生動物園玩”，正向通常會分詞成”我們 在野 生動 物 園 玩”，而反向則會為”我們 在野生 動物園 玩”。正向分詞會分成 6 個詞，而反向為 5 次，因此選擇反向作為最後的分詞結果。

另外，原始 BERT 的模型是使用[MASK]作為 mask token 來替換需要被遮蔽的單詞，但 MacBERT 是將需要被換掉的單詞換成語意接近的詞彙。由於[MASK]在後續 fine-tuning 的時候根本不會出現，因此用相近語意的詞彙做替換能更好的貼近下游的任務並在訓練時就縮小訓練前後的差距。

網媒所碩一 R10944028 蔡易儒

ADL HW2 report

2. Answer Span(1%):

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

讓 tokenizer 回傳 offset_mapping 當作每個 token 對應原本 char 的起始處與結尾處，接著用兩個 pointer，token_start_index 與 token_end_index，從兩端往內分別搜索出原本 label 的 start_char 與加上 label 長度的 end_char。

詳細的作法是從頭依序檢查每個 token 的 start_index 與答案的 start_index，若小於等於的話，就繼續往下個字找，一直到 token 的 start_index 大於答案的 start_index，才停下來，而 token_end_index 也是相同的方法只是從尾巴依序檢查 token 的 end_index 是否小於答案的 end_index，滿足才停下來。得到的 start_index 與 end_index 就是答案的 token_start_index 與 token_end_index。

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?

先分別取 start_index 與 end_index 機率最高的 n 個選項選出來，接著用暴力法把這些 start_index 與 end_index 的組合列出來後，用兩者的 logit score 相加起來當作 sorting 的依據(因為應該有取過 exponential，所以其實是機率相乘)，此過程中需要將不可能為答案的選項直接忽略掉，如長度(end_index-start_index)小於 0 或大於 max_answer_length，也不用考慮 out of scope 的答案，如 start_index 或 end_index 超出原本 offset_mapping 的範圍或無法 map 到的 char，通過檢查後且機率最大的組合就是我們最後的答案了。

ADL HW2 report

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

1. Describe (2%)

   a. your model (configuration of the transformer model)

   使用 chinese-macbert-base 與 chinese-macbert-large，左邊為 context selection，右邊為 question-answering。


```
{
  "_name_or_path": "hfl/chinese-macbert-base",
  "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",
  "torch_dtype": "float32",
  "transformers_version": "4.17.0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
}
```

```
{
  "_name_or_path": "hfl/chinese-macbert-large",
  "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": 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": 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
}
```


   b. performance of your model.

   Eval context selection accuracy: 0.9684

   Eval QA exact match: 84.6128

   Eval QA F1: 84.6128

   Public test QA exact match: 0.8056

   Private test QA exact match: 0.81029

ADL HW2 report

c. the loss function you used.

使用 Cross Entropy loss 與 L2 regularization， $\hat{y}$為 predict 的結果，y為 ground truth label，N 為 label 的數量，λ為 regularization 的參數。

$$Loss = CrossEntropyLoss(\hat{y}, y)$$

$$= \sum_{n=1}^{N} -y_n \log\left(\frac{\exp(\hat{y}_n)}{\sum_{c=1}^{num_{class}} \exp(\hat{y}_n)}\right) + \lambda * \left||w|\right|_2^2$$

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

Optimizer: AdamW

Learning rate: 5e-5

Batch size: 2*4*32 = 256

(per_device_batch_size*device#*gradient_accumulation)

LR_scheduler_warmup_ratio: 0.2

ADL HW2 report

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

a. your model (configuration of the transformer model)

均使用 bert-base-chinese


```
{
  "_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
}
```


b. performance of your model.

Eval context selection accuracy: 0.96444

Eval QA exact match: 0.79462

Eval QA F1: 0.79462

Public test QA exact match: 0.75226

Private test QA exact match: 0.76332

c. the difference between pretrained model

由於 macbert-base 與 macbert-large 僅有參數的差異，其概念是大同小異，因此此部分會著重於 MacBERT 與 BERT 的差異進行說明。

在預處理方面，與 BERTtokenizer 是將一個字切成 subword 不同，

ADL HW2 report

MacBERT 則是將含有語意的基本單位作為分詞的依據，而若要將一個 token 進行遮蔽的動作的話，則會將整個詞都遮掉，另外有特別去分配不同長度詞的比例，從 unigram、bigram 一直到 4-gram，換成 mask 的比例是固定的。

原始 BERT 的模型是使用[MASK]，來替換需要被遮蔽的單詞，雖然上述都說是換成 mask，但其實 MacBERT 是將需要被換掉的單詞換成語意接近的詞彙。由於[MASK]在後續 fine-tuning 的時候根本不會出現，因此用相近語意的詞彙做替換能更好的貼近下游的任務並在訓練時就縮小訓練前後的差距。

而超參數的部分，兩者大致滿接近的，只有 optimizer 的選擇不太一樣，BERT 是使用 AdamW，而 MacBERT 是使用 LAMB(Layer-wise Adaptive Moments optimizer for Batch training)，訓練資料的部分 MacBERT 也多了 wiki 的 extended data，雖然 word 的數量變多許多，但 vocab 的數量則維持不變。

網媒所碩一 R10944028 蔡易儒

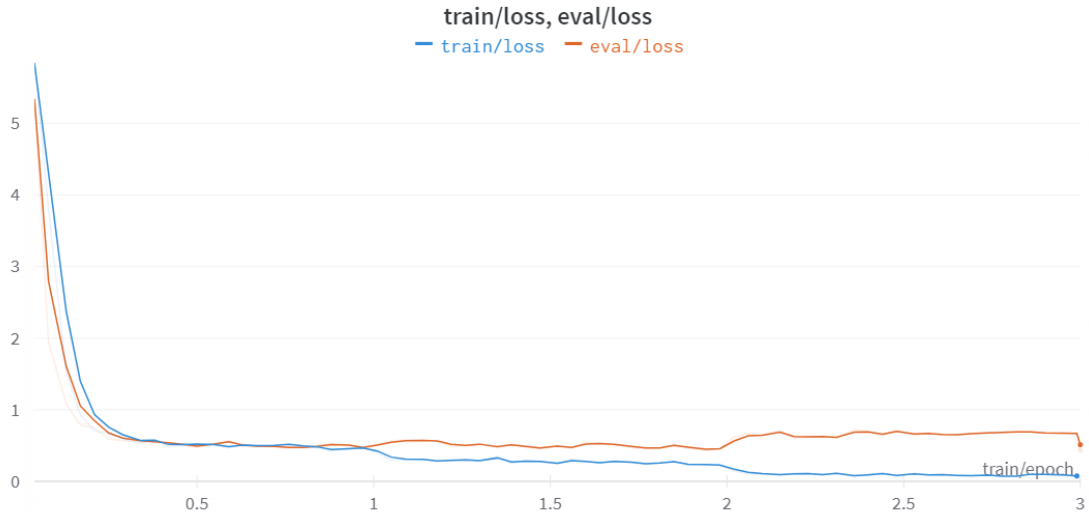
ADL HW2 report

Q3: Curves

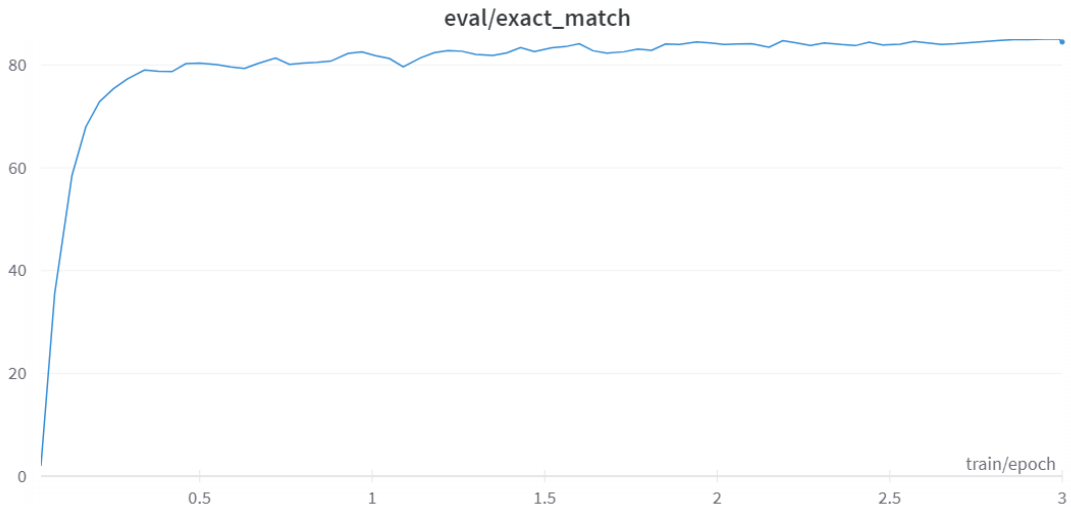
1. Plot the learning curve of your QA model

a. Learning curve of loss

由於原版 trainer 應該沒有觀察 evaluation loss 的功能，因此我參考了他 training 原版的 code (https://github.com/huggingface/transformers/blob/v4.18.0/src/transformers/models/bert/modeling_bert.py#L1849)，來進行 evaluation loss 的製圖。



b. Learning curve of EM

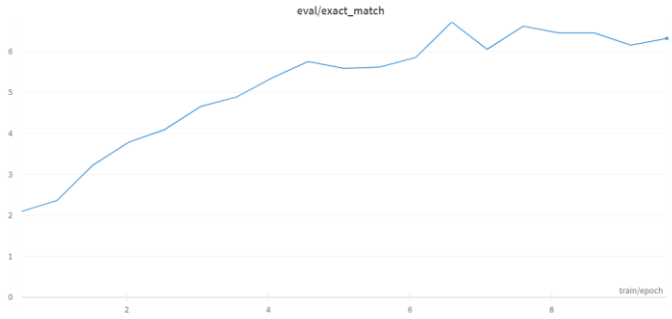


ADL HW2 report

Q4: Pretrained vs Not Pretrained(QA)

1. Configuration


```
{
  "_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.

直接將第二題的 bert-base-chinese 拿來改成不 load pretrained weight 的版本，(AutoModelForQuestionAnswering.from_pretrained 改成 AutoModelForQuestionAnswering.from_config)，因此其表現就十分的差勁。

3. The performance of this model v.s. BERT:

	Eval	Public test	Private test
no pretrained	0.06613	0	0
pretrained	0.79462	0.75226	0.76332

因為模型的大小太大且太過複雜，只用這些資料且只訓練幾個 epoch 的確是沒辦法將模型訓練得太好，與有 pretrained 的完全沒辦法比，不過從以下的 learning curve 可以發現其還在成長中，如果繼續訓練的話可能還是會有機會將其訓練到一定的程度。

ADL HW2 report

Q5: Bonus: HW1 with BERTs

a. your model

兩個都使用 bert-base-cased。Intent selection 改自 text-classification，如左圖；slot tagging 改自 token-classification，如右圖。詳細的程式都有附在上傳的檔案夾中。


```
{
  "_name_or_path": "bert-base-cased",
  "architectures": [
    "BertForSequenceClassification"
  ],
  "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",
  "problem_type": "single_label_classification",
  "torch_dtype": "float32",
  "transformers_version": "4.17.0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 28996
}
```

```
{
  "_name_or_path": "bert-base-cased",
  "architectures": [
    "BertForTokenClassification"
  ],
  "attention_probs_dropout_prob": 0.1,
  "classifier_dropout": null,
  "finetuning_task": "ner",
  "gradient_checkpointing": false,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "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": "4.17.0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 28996
}
```


b. performance of your model

i. Intent classification

Eval accuracy: 0.95533

Public test accuracy: 0.95111

Private test accuracy: 0.948

ii. Slot tagging

Eval f1: 0.8274

Public test accuracy: 0.84343

Private test accuracy: 0.8419

ADL HW2 report

可以發現 pretrained model 在兩個 task 都有十分好的表現，而且 intent selection 甚至沒有調整參數，用預設的就能輕鬆拿到拿到比 LSTM 第一名還要好的分數。而 slot tagging，則是有稍微調整一下訓練的時間，訓練時間拉長一些才有超過用 LSTM 的分數。

我想應該都是得益於，之前有預訓練過大量的其他資料，並經過

c. the loss function you used

兩者都使用 Cross Entropy loss 與 L2 regularization， $\hat{y}$為預測的輸出，y為 ground truth label，N 為 label 的數量，λ為 regularization 的參數。

$$Loss = CrossEntropyLoss(\hat{y}, y)$$

$$= \sum_{n=1}^{N} -y_n \log\left(\frac{\exp(\hat{y}_n)}{\sum_{c=1}^{num_{class}} \exp(\hat{y}_n)}\right) + \lambda * ||w||_2^2$$

d. The optimization algorithm, learning rate and batch size

兩者都使用，唯 slot tagging 需要訓練比較多 epoch(10)。

Optimizer: AdamW

Learning rate: 5e-5

Batch size: 8*2*8 = 128

(per_device_batch_size*device#*gradient_accumulation)