國立臺灣大學 National Taiwan University

指導教授：陳縕儂
學生 ： 黃泰誠
學號：R09921083

Applied Deep Learning Homework 2

Q1:Data processing

Swag task :

我將這次的中文 QA 任務分成 2 個任務，其一先將課程給的 train.json 檔案轉成 csv 檔來進行 swag 的任務，也就是針對 paragraphs 的相關句子中找出關聯性最高的句子，其資料型態如圖(1)所示，再將 task1 所選出的關聯性最高的句子整理成 json 檔的，並丟入 task2 來進行 squad task。

numb	id	question	context0	context1	context2	context3	context4	context5	context6	label
0	ab3956799	舍本和誰的	所有的恆星	心理學是研	這是一個研	在19世紀雙	超新星爆發	1930年，E	NONE	3
1	2233fa2fa9	在關西鎮的	新竹縣的地	新竹縣人口	然而19814	隨著解嚴以	開發區依照	新竹縣是中	NONE	5
2	d5693f35f4	「有錫兵」	柏拉圖哲學	麥克斯·施	「無錫」-唐代已有	所有權又稱	唐朝，國祚	NONE	2	
3	10b04a025	《方法論》	在歐洲，主	《馬太福音	《後漢書》	古代關於	極限不是數	牛頓和萊布	NONE	0

圖(1)

Squad task:

知道 question 所對應的相關文章後這個任務就只需要專注地找出其對應的答案，我將課程給的 json 檔轉換成如圖(2)的形式，來送入 model 進行 training。


```
"version": "1.1",
"data": [
 {
  "paragraphs": [
   {
    "context": "在19世紀雙星觀測所獲得的成就使重要性也增加了。在1834年，白塞爾觀測到天狼星自行的變化，因而推測有一顆隱藏的伴星；愛德
    "qas": [
     {
      "question": "舍本和誰的數據能推算出連星的恆星的質量？",
      "id": "ab39567999fd376480ac3076904e598e",
      "answers": [
       {
        "text": "斯特魯維",
        "start": 108
       }
      ]
     }
    ]
   }
  ]
 },
```


然而不論是 swag task 或是 squad task 其 tokenize 的方式都是同一種形式，以下進入 tokenizer 的介紹。

1. Tokenizer：

這次在資料的 data process 是參考 bert 原始的分詞方式其主要分成兩個 tokenizer : BasicTokenizer 和 WordpieceTokenizer，另外一個 FullTokenizer 是這兩個的結合：先由 BasicTokenizer得到一個分詞得比較粗略的 token 列表，然後再對每個 token 進行一次 WordpieceTokenizer，得到最終的分詞結果。

BasicTokenizer:

BasicTokenizer 是一個初步的分詞器。對於一個待分詞字符串，流程大致就是轉成 unicode -> 去除各種奇怪字符 -> 處理中文 -> 空格分詞 -> 去除多餘字符和標點分詞 -> 再次空格分詞。

Import unicode:

轉成 unicode 這步對應於 convert_to_unicode(text) 函數，就是將輸入轉成 unicode 字符串。

中文的處理:

處理中文對應於 BasicTokenizer 類的 _tokenize_chinese_chars() 的函式。對於 text 中的字符，首先判斷其是不是中文字符，是的話在其前後加上一個空格，否則原樣輸出。而怎麼判斷一個字符是不是中文呢？

_is_chinese_char(cp) 的函式，cp 指的是碼位，通過碼位來判斷，總共有 81520 個字，詳細的碼位範圍如下

- [0x4E00, 0x9FFF]：十進制 [19968, 40959]
- [0x3400, 0x4DBF]：十進制 [13312, 19903]
- [0x20000, 0x2A6DF]：十進制 [131072, 173791]
- [0x2A700, 0x2B73F]：十進制 [173824, 177983]
- [0x2B740, 0x2B81F]：十進制 [177984, 178207]
- [0x2B820, 0x2CEAF]：十進制 [178208, 183983]
- [0xF900, 0xFAFF]：十進制 [63744, 64255]
- [0x2F800, 0x2FA1F]：十進制 [194560, 195103]

經過這步後，中文被按字分開，用逗點分隔，但英文數字等仍然保持原狀。

之後再去除多餘的字符和標點分詞

我們透過**_run_split_on_punc()**的函式把原始的一句話例如:

【今天天氣不錯】變為【’今’ , ’天’, ‘天’ , ‘氣’ , ‘不’ , ‘錯’】

最後由於中文字並不像英文一般有著子音節的問題，所以 WordpieceTokinezer 就沒有太常使用到除非在問句或文章中有中文的部分，最後透過我在網路上得到的中文 vocab 大越是 21128 的字數來根據相對應的中文字或符號給出相對應的數字如圖(3)所示。

圖(3)

根據圖(3)可以知道我用[CLS],來做為整句個 input 的開頭，並用[SEP]來將問題和文章區隔開來，最後我再用一個[SEP]來宣告整個 input 的結束。

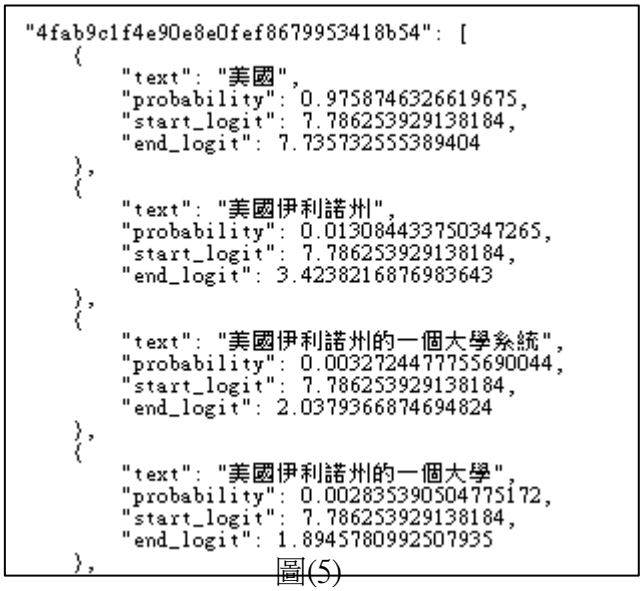
2. Answer Span：

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

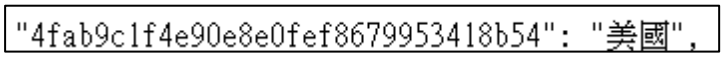
以下看到的”answers”裡面是一個 dictionary 並包裝成 list 我們將 text 和 start 取出並將它建立成 input data 由於我們已經知道 start position 所以可以透過計算字的長度就能知道 end position 在哪裡，如圖(4)所示可以得到 start position 為 190 而【台灣棒球運動珍貴新聞檔案數位資料館之建置】為 20 個字所以將其相加後再減去一。

圖(4)

接著透過 training 訓練出可以判斷句子起始跟終點分布於各個字的機率最後再進行 softmax 來抓出可能的答案，這裡我選擇讓他抓出了 8 個有可能的答案來比較它的 probability 並寫入 nbest_predictions.json 如圖(5)所示，根據 nbest_predictions.json 所得到的 probability 選出最高的做為最後輸出的答案如圖(6)所示。



圖(5)



圖(6)

Q2: Modeling with BERTs and their variants

1. **Describe**

   a. my model

   RoBERTa-wwm-ext_Chinese_large

   選擇這個 model 作為通過 baseline 的原因如下:

   1. WWM 為(Whole Word Masking)全詞遮罩技術，是谷歌在 2019 年 5 月 31 日發布的某一 BERT 的升級版本，主要更改了原預訓練階段的訓練樣本生成策略。
   2. 其訓練數據為 EXT，訓練資料包括:中文維基百科，其他百科，新聞，問答等數據，總詞數達 5.4B。
   3. Large 的神經層數較複雜，共有 24 個 layer 而每個 larye 共有 1024 個神經元，經過長時間的訓練可以讓 model 更精準。

   b. performance of your model

   em : 78.927

   f1 : 85.43

   ```
   {'count': 3526, 'em': 0.7892796369824163, 'f1': 0.854328441540328}
   ```

   c. the loss function you used

   CrossEntropyloss

   ```
   def forward(self, input_ids, token_type_ids=None, attention_mask=None, start_positions=None, end_positions=None):
       sequence_output, _ = self.bert(input_ids, token_type_ids, attention_mask, output_all_encoded_layers=False)
       logits = self.qa_outputs(sequence_output)
       start_logits, end_logits = logits.split(1, dim=-1)
       start_logits = start_logits.squeeze(-1)
       end_logits = end_logits.squeeze(-1)

       if start_positions is not None and end_positions is not None:
           # If we are on multi-GPU, split add a dimension
           if len(start_positions.size()) > 1:
               start_positions = start_positions.squeeze(-1)
           if len(end_positions.size()) > 1:
               end_positions = end_positions.squeeze(-1)
           # sometimes the start/end positions are outside our model inputs, we ignore these terms
           ignored_index = start_logits.size(1)
           start_positions.clamp_(0, ignored_index)
           end_positions.clamp_(0, ignored_index)

           loss_fct = CrossEntropyLoss(ignore_index=ignored_index)
           start_loss = loss_fct(start_logits, start_positions)
           end_loss = loss_fct(end_logits, end_positions)
           total_loss = (start_loss + end_loss) / 2
           return total_loss
       else:
           return start_logits, end_logits
   ```

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

   Optimization algorithm: Adam

   Learning rate: 5e-5

   Train batch size: 1

   Validation batch size: 20

2. Try another type of pretrained model and describe

a. my model

BERT base

選擇這個 model 的原因如下:

1. 起初為先了解如何使用 bert 來完成 nlp 任務
2. 其 layer 數量和每個 layer 的神經元數量都遠小於其他 model 故可以做各種參數的調整來理解 bert 再 max length 以及 doc_stride 的調整。

b. performance of your model

em: 64.8

f1: 70.5

c. the loss function you used

CrossEntropyloss

```
def forward(self, input_ids, token_type_ids=None, attention_mask=None, start_positions=None, end_positions=None):
    sequence_output, _ = self.bert(input_ids, token_type_ids, attention_mask, output_all_encoded_layers=False)
    logits = self.qa_outputs(sequence_output)
    start_logits, end_logits = logits.split(1, dim=-1)
    start_logits = start_logits.squeeze(-1)
    end_logits = end_logits.squeeze(-1)

    if start_positions is not None and end_positions is not None:
        # If we are on multi-GPU, split add a dimension
        if len(start_positions.size()) > 1:
            start_positions = start_positions.squeeze(-1)
        if len(end_positions.size()) > 1:
            end_positions = end_positions.squeeze(-1)
        # sometimes the start/end positions are outside our model inputs, we ignore these terms
        ignored_index = start_logits.size(1)
        start_positions.clamp_(0, ignored_index)
        end_positions.clamp_(0, ignored_index)

        loss_fct = CrossEntropyLoss(ignore_index=ignored_index)
        start_loss = loss_fct(start_logits, start_positions)
        end_loss = loss_fct(end_logits, end_positions)
        total_loss = (start_loss + end_loss) / 2
        return total_loss
    else:
        return start_logits, end_logits
```

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

Optimization algorithm: Adam

Learning rate: 5e-5

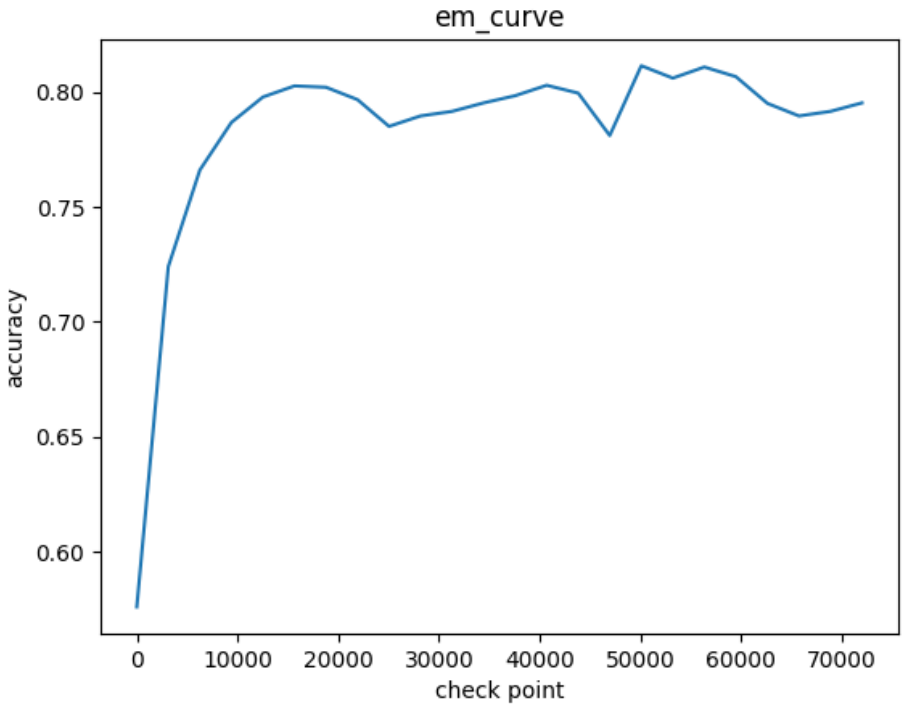
Train batch size: 8

Validation batch size: 32

Q3: Curves

1. Plot

a.



b.