1. data preprocessing:(code 中的幾個重要的變數)

get_paragraph_QA()函數是為了要得到以下幾個變數

- train_paragraph:每筆資料以一個換行作為分隔，而一筆資料中又以<s>,</s>作為每一句的區分，所以這個變數是一個二維的 list。而在每一句中我會去掉<s></s>這兩個符號。train_paragraph[i][j],i=0~資料筆數,j=0~每筆資料總共幾句。
- train_QA:每筆資料個 Q 跟 A。train_QA[i][j],i=0~資料筆數,j=0,1，train_QA[i][0]代表第 i 筆資料的 Q，train_QA[i][1]代表第 i 筆資料的 A。

get_data()函數是為了要得到以下幾個變數

- train_vector:因為我用的 pretrain model 的關係，我參考了 https://huggingface.co/bert-large-uncased-whole-word-masking-finetuned-squad#pretraining 這篇的 pretrain 設置，bert 的 input sequence length 我這邊設為 384，所以這個變數會把每一筆資料中的每一句不斷串接起來，每一句串接的中間加上'.'(句號)，直到串接長度<=384 為止，每一筆後面的資料就直接去除，所以 train_vector 為一個 list，len(train_vector)=資料筆數，train_vector[i]為第 i 筆要直接 input 進 bert model 的資料，例如:train_vector:[tensor([1,2,3,...,0,0]),...,tensor([1,26,33,...,54,0])]，每一句都經過了 tokenize 且轉換為 id 型式，後面的數字 0 為長度小於 384 的 padding。
- train_att_mask:bert 的 attention mask，len(train_att_mask)= 資料筆數，例如:train_att_mask :[tensor([1,1,1,...,0,0],...,tensor([1,1,1,...,1,0])]，1 為非 padding 的 token、0 為 padding 的 token。
- train_segment_ids:用來區分 input 句子，例如:train_segment_ids: [tensor([0,0,0,...,1,1,1,1],...,tensor([0,0,0,...,1,1,1,1])]，len(train_segment_ids)=資料筆數，因為我的 input 型式為 [CLS] Question [SEP] Sentence [SEP]，所以前面[CLS] Question [SEP]的部分都用 0 來表示，後面的 Sentence [SEP]都用 1 來表示。
- train_start_pos:標示 answer token 的起始位置，例如:train_start_pos: tensor([33,54,87,1,2,...,333])，len(train_start_pos)=資料筆數，train_start_pos[i]=111 表示第 i 筆資料的 answer 起始位置為 111。
- train_end_pos:同 train_start_pos，start 改 end，起始改結束。
- train_ans: [ans_1,ans_2,...,ans_n] 。
- 同理可推 validation 的部分也一樣。

2. model architecture:

- 使用 BertModel.from_pretrained('bert-large-uncased-whole-word-masking-finetuned-squad')，這個是有在 SQuAD dataset 上 pretrain 過的。
- Output layer design:bert Output 的 hidden state size 我設置為 1024，一整個 sequence 會經過同一個參數的 linear layer，將 size 從 1024 轉為 2，代表

start 跟 end 的數值，在 inference 時 start 用 argmax 得出數值最大的那個位置當作起始位置，同理 end 也是，最後將 start~end 這組 Index 轉為 token 再轉為由字串組成的 answer。

3. training process:

- batch size=3, optimizer 為 AdamW,總共 train 5 個 epochs，一個 epoch 大約需要訓練 3 小時。
- 使用 get_linear_schedule_with_warmup()，從第 0 個 step 開始線性 warmup。
- output layer 在 training 時會將整個 sequence 出來的 start 數值經過 softmax 算 cross entropy loss，同理 end 也會經過 softmax 算 cross entropy loss，最後兩個相加除以 2。
- 我的設置是每 training 完一個 epoch 會在 validation set 上做驗證，並算出 f1 score,EM，然後如果驗證出比之前更高的 f1 score 就會將模型存起來。

4. Another Hyperparameters:

- Droup out rate=0.1

5. evaluation scores:

- 在計算 validation 的 metrics 時也是擷取前面 383 個 token，後面部份去掉來計算。
- 根據這種處理 data 的方式，最好的 f1 score 為 0.7071,EM 為 0.6579。

6. 嘗試其他處理 data 的方法:

- 每一筆 data 取前面 384 token 之外，我還重疊了前一個<s></s>再依序取後面的 384 token，然後對應的 ans 位置都是依序取出的 384 token 第一個出現的 ans 位置，不過用這種方式我式出來效果跟上面說得差不多。

7. Training state:

- {'epoch': 1, 'Train Loss': 1.3932774189204205, 'Train f1': 0.6531433498074382, 'Train EM': 0.5979563213784813, 'Val Loss': 1.372263589696993, 'Val f1': 0.6797758322501495, 'Val EM': 0.6295976390988267, 'Training Time': '3:05:31', 'Validation Time': '0:08:37'}
- {'epoch': 2, 'Train Loss': 0.9391550735978038, 'Train f1': 0.7578279971856975, 'Train EM': 0.7163494289721498, 'Val Loss': 1.5243597483457458, 'Val f1': 0.7014701267699712, 'Val EM': 0.6509753113078529, 'Training Time': '3:04:59', 'Validation Time': '0:08:25'}
- {'epoch': 3, 'Train Loss': 0.6320735994116303, 'Train f1': 0.8212782039696005, 'Train EM': 0.7921759166499699, 'Val Loss': 1.9005161426731194, 'Val f1': 0.7016283573574396, 'Val EM': 0.6491758439501908, 'Training Time': '3:02:48', 'Validation Time': '0:08:11'}
- {'epoch': 4, 'Train Loss': 0.380205636059281, 'Train f1': 0.8629953389823476, 'Train EM': 0.8445001001803246, 'Val Loss': 2.387529248185911, 'Val f1':

0.7021848912769333, 'Val EM': 0.6528467573598215, 'Training Time': '3:01:30', 'Validation Time': '0:08:09'}

- {'epoch': 5, 'Train Loss': 0.17741453099228108, 'Train f1': 0.8868705356572085, 'Train EM': 0.8782809056301343, 'Val Loss': 2.636097402653641, 'Val f1': 0.7071678187484064, 'Val EM': 0.6578852659612755, 'Training Time': '3:02:05', 'Validation Time': '0:08:27'}