Q1：Tokenization

1. text 拿掉無效的字符且將 whitespace character 替換成' '
2. 用空白隔開 CJK character (例如：中文字)產生字串
3. 將字串以空白切成 list
4. 針對 list 裡的 token 做以下動作，並將結果 extend 到一個 tokens_list：
   –將英文字變小寫
   –丟掉 token 裡的重音
   –把標點符號拆開來，例如：'amazing，'會變成['amazing', '，']
5. 若 tokens_list 裡的 token 不存在字典裡，會切成 wordpieces，並用##做前綴
   例如：'amazing'，start 和 end 分別為 token 的開頭以及結尾位置
   Step 1：
   start = 0, end = 7： 'amazing'不在字典裡，end 往前一個
   start = 0, end = 6：'amazin'不在字典裡，end 往前一個
   …
   start = 0, end = 2： 'am'在字典裡，start = end 且 extend 到 sub_tokens_list
   start = 2, end = 7：'azing'不在字典裡，end 往前一個
   …
   start = 2, end = 3： 'a'在字典裡，由於 start>0 需要加##變成'##a'
   start = end 且 extend 到 sub_tokens_list
   start = 3, end = 7： 'zing'不在字典裡，end 往前一個
   …
   start = 3, end = 5： 'zi'在字典裡，由於 start>0 需要加##變成'##zi'
   start = end 且 extend 到 sub_tokens_list
   start = 5, end = 7： 'ng'在字典裡，由於 start>0 需要加##變成'##ng'
   start = end 且 extend 到 sub_tokens_list，那因為 start 沒有比 end 小所以結束
   ➔所以'amazing'變成['am', '##a', '##zi', '##ng']，也就是上面的 sub_tokens_list
   若 token 切成 wordpieces 還是不存在字典裡，那就會用[UNK]替代

Chinese：若字沒有出現在 bert-base-chinese 會變成[UNK]
Ex；長秈米 ='長', '[UNK]', '米'
以下若有##代表一個 word 拆成多個 wordpieces，這樣可以減少沒有出現過的字變成[UNK]的機率
Number：
Ex：10° = '10', '##°' = 10 + °
1786 = '178', '##6' = 178 + 6
English：必須變成小寫，大寫英文字母會變成[UNK]
Ex：amazing = 'am', '##a', '##zi', '##ng' = am + a+ zi + ng

Q2：Answer Span Processing

1. Context 經過 tokenize 後會產生 token_list，逐一掃過各個 token 的字元並設一個變數儲存掃過的數量，掃過一個字元此變數就加一，若此變數等於原始的 start 或 end，其相對 token 在 token_list 的位置即為新的 start 或 end。此外，token 若為'[UNK] '、' [CLS]'或'[SEP]'，必須將其視為一個字元而非五個字元，本身有'## '則忽略'## '不計算。若沒有答案則 start 和 end 都設為 512。Training data 有 99.5%對到答案，沒有對到答案就捨棄不用。
2. 將 model 回傳的 start_scores 和 end_scores 裡 non-context 都設為-inf，經過 softmax 後，在取 start_scores 和 end_scores 裡的最大值的位置當作 start/end position。

Q3：Padding and Truncating

1. The maximum input token length of bert-base-chinese is 512.
2. 使用 tokenizer.prepare_for_model 產生 input_ids、token_type_ids 及 attention_mask，max_length 設成 512，而針對 training data 會將 truncate 後答案被砍掉的丟掉。

Q4：Model

【模型訓練】採用 Adam 當作 optimization alogorithm，learning rate 為 1e-5

傳入參數：

input ids, token type ids, attention mask, answerable, start position, end position

回傳參數：

[cls loss, start and end loss]

模型內將上述傳入參數傳到 transformer 的 BertModel 進行訓練並且回傳 BertModel 最後一層隱藏層的序列(以下用 H 替代)，size：(sequence length, hidden size)。

使用[CLS]的隱藏序列(H[0], 以下用 cls 替代)進行有無答案的判斷：

1. Y = W * cls + B，W 的大小為(hidden size, 1)，B 的大小為(1)
2. 採用BCEWithLogitsLoss當作Loss Function，一次訓練為batch大小的資料量

H 會拆成 start scores 和 end scores(由於處理過程一樣，以下統一用 S 替代)，分別經過 Linear 後跟 start position 及 end position 進行交叉熵的計算：

1. Y = W * S + B，W 的大小為(hidden size, num_labels)，B 的大小為 (num_labels)
2. 採用CrossEntropyLoss當作loss function，一次訓練為batch大小的資料量，start and end loss為start和end各自經過CrossEntropyLoss後取平均。

【模型後處理】

傳入參數：

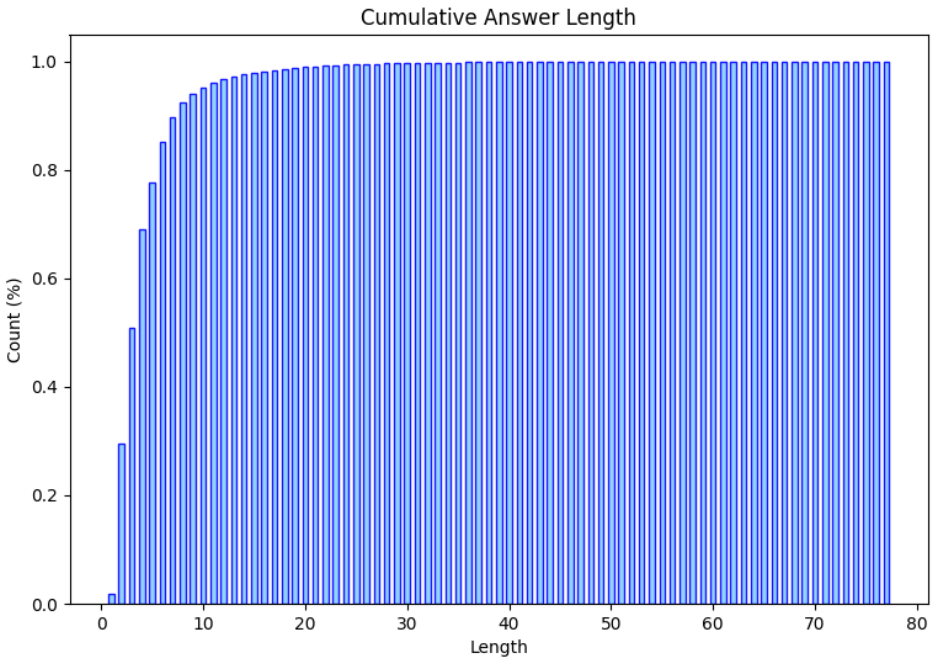
input ids, token type ids, attention mask

回傳參數

cls scores, start scores, end scores

1. cls 內的值大於 threshold 即為有答案，反之無答案。
2. start scores 和 end scores 裡 non-context 都設為-inf，經過 softmax 後，在取 start scores 和 end scores 裡的最大值的位置當作 start/end position。若答案長度超過 60 或 start/end position 為 512 即視為沒有答案。

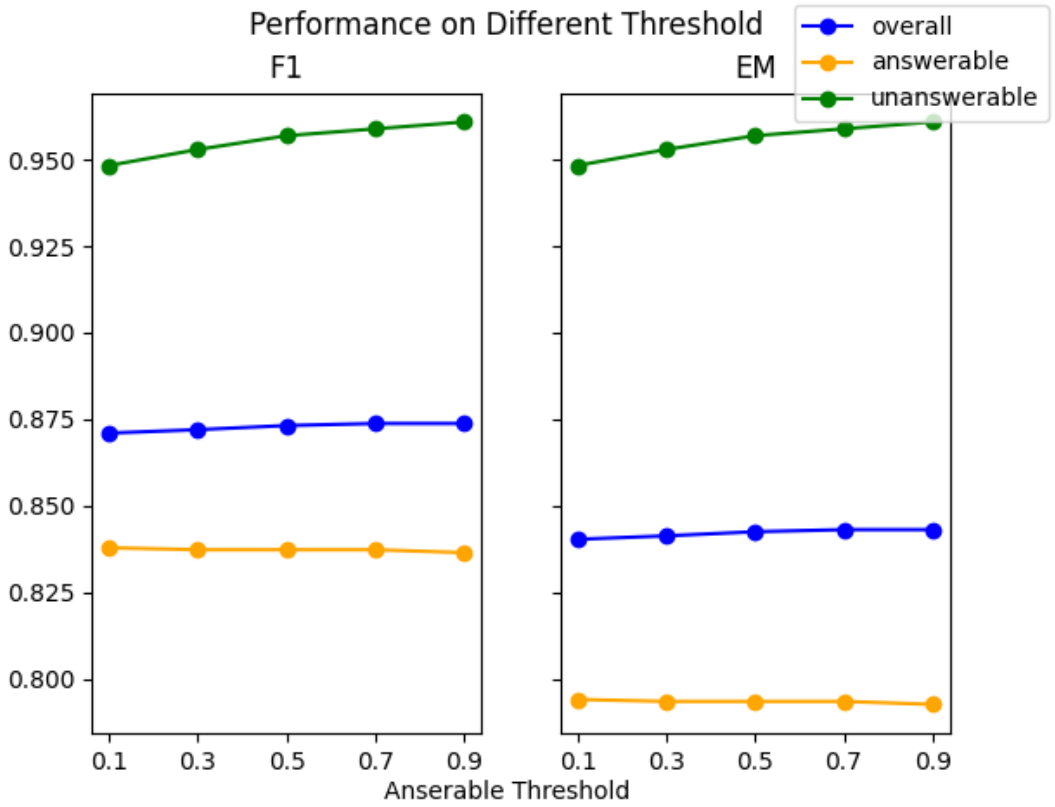
Q5：Answer Length Distribution



從上圖可以發現差不多在長度接近 30 的時候資料數量趨近於全部，這代表所有問題的答案長度大概都介於 1~30，在預測資料進行後處理時可以將答案長度超過 30 的視為沒有答案。

Q6：Answerable Threshold

1. Threshold 設為 0.5。
2.



Q7：Extractive Summarization

1. Context 由 N 個句子組成，斷句及斷詞之後，使用 tokenizer 轉成 id：

Input = $[CLS_0]$ [sent] [zero] $[SEP_0]$ … $[CLS_{N-1}]$ [sent] [N − 1th] $[SEP_{N-1}]$

input_ids = convert_tokens_to_ids (Input)

2. 使用 0 以及 1 製作 token_type_ids

token_type_ids = [0,0,0,0,1,1,1,1,……]，0/1 代表偶/奇數句的 token

3. 經過 bert 後(使用 Binary CrossEntropy 當作 Loss Function)取每個句子的 [CLS]輸出來預測是否為摘要

output_cls = $[CLS_0, CLS_1, \ldots, CLS_{N-1}]$

output_cls 經過 sigmoid 並將值四捨五入，若為 1 代表相對應的句子為摘要