1. (2%) After your model predicts the probability of answer span start/end position, what rules did you apply to determine the final start/end position? (the rules you applied must be different from the sample code)

```
if self.split == "train":
    # Convert answer's start/end positions in paragraph_text to start/end positions in tokenized_paragraph
    answer_start_token = tokenized_paragraph.char_to_token(question["answer_start"])
    answer_end_token = tokenized_paragraph.char_to_token(question["answer_end"])

    # A single window is obtained by slicing the portion of paragraph containing the answer
    mid = (answer_start_token + answer_end_token) // 2
    paragraph_start = max(0, min(mid - self.max_paragraph_len // 2, len(tokenized_paragraph) - self.max_paragraph_len))
    ## [ARRANGE]
    if answer_start_token >  self.max_paragraph_len:
       paragraph_start = min(answer_start_token - self.max_paragraph_len // 2, len(tokenized_paragraph) - self.max_paragraph_len) #不能從0開始
    else:
       paragraph_start = 0
    paragraph_end = paragraph_start + self.max_paragraph_len

    # Slice question/paragraph and add special tokens (101: CLS, 102: SEP)
    input_ids_question = [101] + tokenized_question.ids[:self.max_question_len] + [102]
    input_ids_paragraph = tokenized_paragraph.ids[paragraph_start : paragraph_end] + [102]

    # Convert answer's start/end positions in tokenized_paragraph to start/end positions in the window
    answer_start_token += len(input_ids_question) - paragraph_start
    answer_end_token += len(input_ids_question) - paragraph_start

    # Pad sequence and obtain inputs to model
    input_ids, token_type_ids, attention_mask = self.padding(input_ids_question, input_ids_paragraph)
    return torch.tensor(input_ids), torch.tensor(token_type_ids), torch.tensor(attention_mask), answer_start_token, answer_end_token
```

基於 sample code 的版本，我加上了紅色框框的部分。

我將 answer_start_token 與 self.max_paragraph_len 比較，若 answer_start_token 較大，我就將 paragraph_start 設為「 answer_start_token - self.max_paragraph_len // 2 與 len(tokenized_paragraph) - self.max_paragraph_len 的最小值 」；而若 answer_start_token 較小，則設 paragraph_start 為 0。

2. (2%) Try another type of pretrained model which can be found in huggingface's Model Hub (e.g. BERT -> BERT-wwm-ext, or BERT -> RoBERTa), and describe:

   ● The pretrained model you used: `hfl/chinese-macbert-large`

   ● Performance of the pretrained model you used: Public Score: 0.81605

   ● The difference between BERT and the pretrained model you used (architecture, pretraining loss, etc.):

   MacBERT 是基於 BERT 的改進優化版本，它引入了 MLM as correction, MAC 模型的 Pretrained task，緩解了 Pretrained 下游任務不一致的問題。MLM 模型中，引入了 Mask 標記的技術，但 Mask 會出現在下游任務中。而在 MacBERT 中使用相似詞來取代 Mask 標記。相似詞透過 Synonyms toolkit (Wang and Hu, 2017) 工具進行提取，算法基於 word2vec 相似度機算。同時也引進了 Whole Word Masking (WWM) 和 N-gram masking 技術。當要對 N-gram 進行 Mask 處理時，我們會對 N-gram 裡的每個詞分別查找相似詞。當沒有相似詞可以替換時，就將使用隨機的詞進行替換。另外，MacBERT 主要框架與 NERT 一致，可在不修改程式碼的基礎上直接執行。