AI CUP 2023 秋季賽

隱私保護與醫學數據標準化競賽:

解碼臨床病例、讓數據說故事競賽 競賽報告

隊伍:TEAM_3868

隊員:周孝威(隊長)、陳兆余、蔡欣樺、吳秉諺

Private leaderboard:Rank: 11.00 (9) Task1: 0.7934 (11) Tasl 2: 0.7556 (11)

壹、環境

作業系統: Linux GPU: 1080ti*4 Memory:12GB 程式語言: Python 3.7.16

套件:

transformers

torch

numpy==1.21.2

pandas==1.3.3

datasets==1.18.3

pycountry

https://s3-us-west-2.amazonaws.com/ai2-s2-scispacy/releases/v0.4.0/en_core_sci_sm-0.4.0.tar.gz

https://s3-us-west-2.amazonaws.com

/ai2-s2-scispacy/releases/v0.4.0/en_core_sci_lg-0.4.0.tar.gz

預訓練模型:

roberta-large

來源: https://huggingface.co/roberta-large

額外資料集:

i2b2 NLP data sets

來源: https://portal.dbmi.hms.harvard.edu/

貳、演算方法與模型架構

PART 1.Model Used

我們使用的是huggingface library所提供的transformer model roberta-large作為模型。

RoBERTa是一種在大量英文數據上以自監督方式預訓練的transformers模型。意思是它僅在原始文本上進行了預訓練，完全沒有人為進行任何形式的標註，這也是此模型能夠使用大量公開數據不受限於標註資料量的原因，此模型會通過自動化流程從這些文本中生成輸入和label。

更準確地說，RoBERTa是通過Masked language modeling（MLM）目標進行預訓練的。對於一個句子，該模型會隨機屏蔽輸入中的15%單詞，然後將整個屏蔽的句子通過模型運行，並預測被屏蔽的單詞。這和傳統的recurrent neural networks（RNNs）不同，RNNs通常一次看到一個單詞，也不同於自回歸模型（如GPT），因為自回歸模型會在內部屏蔽未來的label。這使得模型能夠學習到句子的雙向表示。

如此，模型就學到了英語語言的內部表示，然後可以用這種表示提取對下游任務有用的特徵。例如，如果你有一個數據集，內有標註好的句子，你可以使用RoBERTa模型生成的特徵向量作為輸入，然後訓練一個標準的分類器。

而RoBERTa與另一個常用的自然語言處理模型BERT差異在於RoBERTa對訓練數據採取了更多的自我監督策略。並且，RoBERTa移除了BERT中的"Next Sentence Prediction"任務，並使用更大的小批量大小和更長的訓練步數。總的來說，RoBERTa可以視為對BERT的一種改進和優化，通過調整預訓練的目標、文本處理方式、訓練方式等方面，提高了模型的性能和泛化能力。然而，兩者仍然共享相同的基本架構，都屬於Transformer-based的自然語言處理模型。

PART 2.Model Parameter

我們主要的模型參數如下所示:

```
{ "post_process":"argmax",
```

```
    "model_name_or_path":"roberta-large",
    "task_name":"ner",
    "notation":"BILOU",
    "ner_types":["DOCTOR", "DATE", "IDNUM", "MEDICALRECORD",
"PATIENT", "HOSPITAL", "TIME", "DEPARTMENT", "CITY", "ZIP",
"STREET", "STATE", "ORGANIZATION", "AGE", "DURATION", "SET",
"PHONE", "LOCATION-OTHER", "COUNTRY", "URL", "ROOM",
"USERNAME", "PROFESSION", "FAX", "DEVICE", "EMAIL", "BIOID",
"HEALTHPLAN"],
    "max_length":250,
    "preprocessing_num_workers":6,
    "per_device_train_batch_size":2,
    "per_device_eval_batch_size":2,
    "gradient_accumulation_steps":8,
    "weight_decay":0.01,
    "learning_rate":5e-5,
    "num_train_epochs":10,
    "adam_epsilon":1e-06   }
```


參數說明則如下所示:

post_process: "argmax"：使用argmax作為後處理的方法，即在預測中選擇概率最大的標籤。

model_name_or_path: "roberta-large"：使用RoBERTa模型的大型預訓練模型。

task_name: "ner"：執行命名實體識別（NER）的任務。

notation: "BILOU"：使用BILOU（Begin, Inside, Last, Outside, Unit）標記體系進行NER標註。

ner_types: [...]：列舉NER任務中可能的實體類別。

max_length: 250：輸入序列的最大長度。

preprocessing_num_workers: 6：用於數據預處理的並行工作程序數量。

per_device_train_batch_size: 2：每個訓練設備的批次大小。

per_device_eval_batch_size: 2：每個評估設備的批次大小。

gradient_accumulation_steps: 8：累積梯度的步驟數，有助於處理較小的批次大小。

weight_decay: 0.01：權重衰減的強度。

learning_rate: 5e-5：初始學習速率。

num_train_epochs: 10：訓練的總期數。

adam_epsilon: 1e-06：Adam優化器的epsilon參數。

參、創新性

Postprocess

我們注意到使用AI來做時間資訊正規化的效果還有一些可以補足之餘，所以我們在後處理下了一番功夫，決定採用物理方式去擷取所有有關DATE、TIME、DURATION、SET的所有可能，主要是使用re正規化函式為主軸去做延伸，將所有觀察到的在文本中的資訊歸納成幾種規則闡述。

DATE以dp = r'(\d{1,2}[/.-]\d{1,2}[/.-]\d{2,4})|(\d{2,4}[/.-]\d{1,2}[/.-]\d{1,2})|(\d{1,2}[/.-]\d{1,2}[/.-]\d{1,2})'的切法為主，當然有可能混雜一些英文月份，我們有在例外發生時做處理，所以即可解決。

TIME以tp = r'\d+[:.]*\d*[ ]*[pPaA]+[.]*[mM]+[.]*|\d+[:.]*\d*'的切法為主，當然在TIME類別也有DATE的格式我們沿用上述DATE的正規劃格式做處理也能有不錯的效果。

DURATION和SET則以窮舉的方式將所有有包在訓練和驗證集的所有可能寫成Function對照表去做轉換，畢竟這兩個的類別較少，基本上有預測到子任務2所出來的結果都不會太差。

所以將我們predict完的.txt文本答案丟入我們寫的regex_all_process.ipynb檔案即能獲取最後也完成子任務 2：時間資訊正規化的最後結果。

肆、資料處理

PART 1. Preprocess

通過執行preprocess.py，我們對資料進行前處理，目的是生成用於訓練或預測的數據集。首先，程式透過迭代文件中的每一行，提取文檔的相關信息，如文本、metadata、文檔 ID、標籤等。再根據模式（mode）是"train"還是"predict"，選擇處理標注的方式。在 "train" 模式下，提取文檔中的標註信息；在 "predict" 模式下，標註信息為空。接著，通過調用 _dataset.get_tokens方法，處理文檔的tokens，此過程會通過Spacy中的切詞再加上re所提供的自訂切詞做切詞處理，而後調用 _sentence_dataset.get_sentences方法，處理每個句子，返回當前句子的信息以及前後文。最後生成一個格式化的字典，其中包含句子的信息，例如tokens、標籤、文檔 ID 等。返回這些處理後的句子，形成一個可迭代的數據集，用於訓練或預測 NER 模型。

PART 2. Postprocess

通過執行merge_answer.ipynb和regex_all_process.ipynb，我們對資料進行後處理，目的是防呆與對預測到的entity做正則化及優化。merge_answer.ipynb主要是將我們以不同切詞方式(formal、all_split、lower_continue)處理資料所訓練成的模型預測結果依照各模型的優勢做預測結果的合併(詳見後續章節)。至於regex_all_process.ipynb，程式開始時，通過pd.read_csv讀取預測的答案文件，創建了一個Pandas dataframe，其中包含了文件ID、標籤、開始和結束位置、entity等信息。然後，通過讀取文件夾中的原文本文件，獲取每個文件的內容，並使用正則表達式從文本中提取日期、時間等信息，並將其添加到dataframe中。接著，對數據進行了一系列處理，包括去除重複資料、處理時間信息、對日期、時間、國家、持續時間等標籤進行進一步的正規化處理和優化。最後，將處理後的數據保存為一個新的答案文本文件，這就是我們最後會上傳的答案文件。

伍、訓練方式

PART 1. Preprocess Step

將主辦方所提供的訓練集做前處理，將所有醫療病例轉換成制定的格式，以下為一個文本的範例

{"text":"文本內容",

"meta":{"note_id": "1"},

"spans": [{"id":"0", "start": 40, "end": 50, "label": "DATE"},.....]}

緊接著將做好上述處理的資料丟進preprocess.py進行前處理，此程式幫我們對這些文本進行切詞和詞性標註label，切詞的規則皆可自己設計，因為沒有一個最完美的切詞，當然在train有標註的情況下可以看答案去訓練，但我們沒有這樣做，

而是通過Spacy中的切詞再加上re所提供的自訂切詞來決定，這樣即使是testing也可以用此邏輯通用，標註我們是使用muti-span的BILUO去做詞性標註，為的是更能分類出每個切出來的詞之間的重要性，跑完之後即可得到我們要丟進roberta-large的前處理train.jasonl檔案，當然test或validation資料也需要跑此前處理。

切詞邏輯自訂義位於程式preprocess.py中
DatasetCreator(L2)->get_tokenizer(L67)->SpacyTokenize(L4)->SpacyTokenizer

PART 2. Training Step

首先在訓練步驟有兩個檔案，分別是train.py和train_AI_cup.json，主要是要設定.json中的參數設置，重要參數如下

"per_device_train_batch_size":2,

"per_device_eval_batch_size":2,

"gradient_accumulation_steps":8,

"weight_decay":0.01,

"learning_rate":5e-5,

"num_train_epochs":10,

"adam_epsilon":1e-06,

"warmup_ratio":0.3,

"logging_steps":5,

還有讀取前處理的檔案路徑必須準確，我們使用hugging face所提供的模型roberta-large跑我們的訓練，跑完之後設定每個epoch存的ckpt檔案，可以使用validation去計算precision、recall、F1-Score來確認模型的穩定性。

PART 3. Predict Step

接下來在預測步驟有兩個檔案，分別是predict.py和predict_AI_cup.json，主要是要設定.json中的參數設置，重要參數如下

"model_name_or_path":"想使用ckpt檔案的路徑位置",

"output_AI_cup_prediction_file":"輸出答案的位置",

"deid_strategy" : "ai_cup",

跑完上述程式即可獲得最終子任務 1：病患隱私資訊擷取的答案。

PART 4. PostProcess System

最後這一步為PostProcess，執行檔案為 regex_all_process.ipynb 以及merge_answer.ipynb，功能分別為產生正規格式以及合併預測結果，合併的候選者我們會挑 eval 分數最高的 formal 並和相對應的lower_continue及all_split決定性的進行Merge的動作，至於如何合併呢，我們是取一個文本的start為基準，並將formal和lower_continue及all_split進行entity的長度比較，取entity較長的當作最後答案，我們認為短的預測會擷取不完全資訊，所以取了其他預測較好的結果來讓我們最終預測結果更完美。

陸、分析與結論

分析:

Training Step

我們在訓練時發現了切詞對模型的重要性，因為其時在傳統無論是各種BERT的Tokenzier都不能幫我們把真正想要切出來的詞會所作切的動作，所以衍伸出我們在這次比賽上不斷想使用如何的切詞邏輯對這次醫療文本的效果是最好的，經過一番研究我們統整出了三種切詞的規律。

1.formal:以標點符號做切詞

2.all_split:實質上的全切，以Character為單位

3.lower_continue:只有小寫不全切，其他都全切

以下為validation所有label的平均分數在各epoch之比較表格:

	formal

評分方式	precision	recall	f1
epoch4	0.97873	0.98182	0.98027
epoch5	0.9834	0.98318	0.98329
epoch6	0.98664	0.98374	0.98519

	formal_all_split		
評分方式	precision	recall	f1
epoch4	0.9823	0.8894	0.9335
epoch5	0.981	0.9713	0.9761
epoch6	0.9864	0.978	0.9822

	formal_all_split_lower_continue		
評分方式	precision	recall	f1
epoch4	0.9791	0.9765	0.9778
epoch5	0.9834	0.9811	0.9823
epoch6	0.9876	0.9814	0.9845

從上可分別看出formal和formal_all_split_lower_continue的F1比較高，確實在最後我們也是使用這兩中方法所取得的答案來做最後上傳。

Postprocess Step

我們發現在新增了後處理和單純模型預測直接出來的效果還是有差異的，因為後處理我們寫了了許多去noise的程式讓我們預測的結果變得更加理想。

以下是無後處理和有後處理的分數比較:

將Train data 13%為testing的分數

	Task1 SCORE
Original	0.7152461
Add_Postprocess	0.861572

True testing:

	Task1 SCORE
Original	0.6754673
Add_Postprocess	0.8196778

總結:

看起來加上了後處理對這次比賽的子任務一是有實質性的幫助的，並且就算在最後經賽結果分數只有0.7933558獲得了第九名，但加上TIME正規化的規則去將文本中搜尋取出，我們的分數就上升到了Task1: 0.8197 (8) Tasl 2: 0.8523(2)的成績，是直接上到本次競賽第二名的，也證明我們所做Postprocess對這次比賽是有極大貢獻。

我們證明:

- 切詞方法對訓練模型的重要性
- Postprocess的對此次競賽的重要性

柒、程式碼

GitHub連結：https://github.com/vicaasas/AI_CUP_deidentification

捌、使用的外部資源與參考文獻

[1]obi-ml-public/ehr_deidentification
https://github.com/obi-ml-public/ehr_deidentification

[2]roberta-large
https://huggingface.co/roberta-large