



AICUP 2023 秋季賽巡迴課程

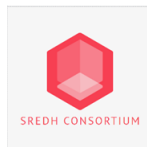
隱私保護與醫學數據標準化競賽

解碼臨床病例、讓數據說故事

線上課程影片



- 戴鴻傑教授團隊
- 國立高雄科技大學 電機工程系
- 亞洲大學生物資訊與醫學工程學系
- 澳洲 SREDH 協會



報告人：李佑謙(Ovien)



Personal introduction

聯絡方式 : ken19980727@gmail.com



李佑謙 (Ovien Lee)

□ Education :

國立高雄科技大學電機工程系-碩士 2020 – 2022

國立高雄科技大學電機工程系-學士 2016 – 2020

□ Experience :

華碩電腦股份有限公司-智慧機器人_BU Aug 2022 – Now

技術工程師-智悠科技股份有限公司「AI.FREE Team」 Feb 2021 – Oct 2021

實習生-信邦電子股份有限公司 July 2019 – Aug 2019

□ AICUP Competition :

AICUP 2020 醫病訊息決策與對話語料分析競賽 秋季賽：醫病資料去識別化 前標 (21/469) Dec 2020

AICUP 2020 台灣高經濟作物 - 愛文芒果影像辨識正式賽 完賽 Sep 2020

AICUP 2019 人工智慧論文機器閱讀競賽之論文標註 前標 (33/531) Dec 2019



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

- 病患隱私資訊擷取子任務
- 時間資訊正規化子任務

競賽介紹



Competition introduction



Medical Report

Task1

病患隱私資訊擷取

In December 1903 DATE the Royal Swedish Academy of Sciences ORG awarded Marie PERSON and Pierre Curie PERSON, along with Henri Becquerel PERSON, the Nobel Prize in Physics WORK_OF_ART.

Task1 Score

Task2

時間資訊正規化

02/09/2014 at 11:42 => 2014-09-02T11:42

Task2 Score



專有名詞辨識



Medical Report



De-identified PHIs

Barack Obama Person the 44th President of the United States Title, was born in Honolulu, Hawaii Location. He graduated from Columbia University Org and Harvard Law School Org. In 2009 Date, Obama was elected as the first African American Ethnicity President of the United States Location. During his presidency, Obama implemented the Affordable Care Act Law and strengthened diplomatic relations with Cuba Location. He served two terms in office before being succeeded by President Donald Trump Title in 2017 Date.



De-identification Medical Report

GDPR規定電子病歷需經過
去識別化才可二次使用
去識別化欄位由HIPAA訂定

一般資料保護規則 (GDPR)
健康保險可攜與責任法 (HIPAA)
受保護的健康資訊 (PHI)



Task1-受保護健康訊息項目

PHI 類別	類型定義	範例
姓名	病患名、醫師名、人名	John Doe, Dr. Max, Mr. Smith
職業	無	lawyer, teacher
地點	診間號、部門、醫院、組織、街、城市、州、國家、區號、其他	peri-operative unit-pow, macquarie ward-rhw,12 abc street
年齡	無	23, 98
日期	日期、時間、週期、頻率	24/12/1987, September 26th
聯絡方式	手機號碼、傳真、電子郵件信箱、網址、網際網路協定位址	+61-421123456、abc@gmail.com、194.223.1.1
識別符	社群安全碼、醫療紀錄號碼、健康計畫號碼、帳戶、證照號碼、車牌裝置號碼、生物識別碼、識別碼	Mrn : 9174338 Id number : 12rl500257
其它	無	

PHI 類別	類型定義	競賽提供的資料集中的類型名稱
姓名	病患名 醫師名 使用者名稱	PATIENT DOCTOR USERNAME
職業	無 診間號 部門 醫院 組織 街	PROFESSION ROOM DEPARTMENT HOSPITAL ORGANIZATION STREET
地點	城市 州 國家 區號 其他	CITY STATE COUNTRY ZIP LOCATION-OTHER
年齡	無 日期 時間 週期 頻率	AGE DATE TIME DURATION SET
日期	手機號碼 傳真 電子郵件信箱 網址 網際網路協定位址 社群安全碼 醫療紀錄號碼 健康計畫號碼 帳戶 證照號碼 車牌 裝置號碼 生物識別碼 識別碼	PHONE FAX EMAIL URL IPADDR SSN MEDICALRECORD HEALTHPLAN ACCOUNT LICENSE VECHICLE DEVICE BIOID IDNUM
聯絡方式	無	OTHER
識別符		
其它		



Task1-資料集介紹

一份 Medical Report 會有多個 Annotation Tag

Medical Report xxx.txt

Episode No: **14K068256G**
9425477.STG

RODWELL, ROBERT WILLIAM
Lab No: **14H02780**
"STOCKDALE" 653 MONAGHAN RD
MANILDRA NSW 2865
Specimen: Tissue
D.O.B: **27/07/1950**
Sex: M
....

Medical Report Annotation Info				
檔案名稱	PHI類別	起始位置	結束位置	隱私資訊內容
xxx.txt	IDNUM	13	23	14K068256G
xxx.txt	MEDICALREC ORD	24	35	9425477.STG
xxx.txt	PATIENT	37	60	RODWELL, ROBERT WILLIAM
xxx.txt	IDNUM	71	79	14H02780
xxx.txt	STREET	80	107	"STOCKDALE" 653 MONAGHAN RD
xxx.txt	CITY	108	116	MANILDRA
xxx.txt	STATE	118	121	NSW
xxx.txt	ZIP	123	127	2865



Task1-評分方式

每一個病患隱私資訊 (Protected Health Information , 簡稱 PHI) 包含如下三個屬性：

- PHI 在文本中的 (1) 起始位置和 (2) 結束位置以及
- PHI 的 (3) 類別資訊

基於上述三個屬性，本子任務中定義：

- True Positive (簡稱 TP) ：參賽團隊系統所辨識的 PHI 在文本中的起始與結束位置及其預測的類別完全符合人工標注的 PHI 對應的三個屬性；
- False Positive (簡稱 FP) ：系統辨識的 PHI 的任何一種屬性 (起始、結尾或類別) 不符合任何人工標注的 PHI 結果；
- False Negative (簡稱 FN) ：人工標注的 PHI 找不到完全匹配的系統辨識結果。

基於上述定義，本子競賽採用 Precision 、 Recall 和 F1-Measure 來評估，並以 Macro-F1-Measure 做為本子任務的排名指標。



Task2-時間資訊正規化

子類別	範例 (需要標註的文字以粗體標記)	正規化格式
DATE	D.O.B: 16/01/1941 (JL/ta 20/11/13)	1941-01-16 2013-11-20
TIME	Collected: 02/09/2014 at 11:42 .. A/Prof E Salisbury at 9:30am on 18/3/14. Dr N Lambie at 9:16am on the 18th of September 2013	2014-09-02T11:42 2014-03-18T09:30 2013-09-18T09:16
DURATION	... whilst 20weeks pregnant in 2013 Lung cancer resected two weeks ago.	P20W P2W
SET	... tested twice with positive controls;	R2



Task2-資料集介紹

D.O.B: 09/08/2957

Sex: F

Collected: 14/02/3014 at 11:42

Location: 3 ARRIETTA CLOSE-POW

DR AADLAND ABRAHAM

CLINICAL:

Lymphoma. Duodenal uptake on PET scan.

Result required for multidisciplinary meeting
on Friday.

Medical Report Annotation Info

檔案名稱	PHI類別	起始位置	結束位置	隱私資訊內容	時間正規化資訊
xxx.txt	DATE	8	18	09/08/2957	2957-08-09
xxx.txt	TIME	38	57	14/02/3014 at 11:42	3014-02-14T11:42
xxx.txt	DATE	208	214	Friday	3014-02-18



ChatGPT Zero Shot Learning



這裡有一份醫學病例報告

Episode No: 09F016547J
091016.NMT

SIZAR, HOWARD
Lab No: 09F01654
Runford
RENMARK TAS 5084
Specimen: Tissue
D.O.B: 24/8/1993
Sex: M
Collected: 28/08/2013 at 08:26
Location: St Vincent-BATLOW/ADELONG MULTI PURPOSE SERVICE
DR JAXON AL-KARSTEN
Distribution: FILE-COPY

請幫我把可能跟病人有關的隱私資訊挑出來



根據提供的醫學病例報告，以下是可能與病人有關的隱私資訊，我已經將其挑出：

1. 患者姓名：SIZAR, HOWARD
2. 出生日期 (D.O.B)：24/8/1993
3. 性別：M (男性)
4. 住址：Runford, RENMARK, TAS 5084
5. Episode No: 09F016547J
6. Lab No: 09F01654
7. 送檢日期 (Collected)：28/08/2013 at 08:26
8. 收檢地點 (Location)：St Vincent-BATLOW/ADELONG MULTI PURPOSE SERVICE
9. 主治醫生：DR JAXON AL-KARSTEN
10. Distribution: FILE-COPY

這些資訊屬於私人醫療資訊，應當受到保護，並避免在未經授權的情況下進行公開或分享。
請確保遵守相關的隱私法律和政策，以保護患者的隱私。



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

- 病患隱私資訊擷取子任務
- 時間資訊正規化子任務

資料處理



資料格式

Medical Report

Episode No: 09F016547J[IDNum]

091016.NMT[MedicalReport]

SIZAR, HOWARD[PATIENT]

Lab No: 09F01654[IDNum]

Runford[Street]

RENMARK[City] TAS[State] 5084[ZIP]

Specimen: Tissue

D.O.B: 24/8/1993[Date]

Sex: M

Collected: 28/08/2013 at 08:26[Time]

檔名	PHI類型	報告中起始位置	報告中結束位置	隱私資訊內容
10	IDNUM	14	24	09F016547J
10	MEDICALRECORD	25	35	091016.NMT
....				
10	DATE	122	131	24/8/1993 1993-08-24

Annotation File

時間正規化的欄位

一組標註有 5 個欄位，如果有時間正規化則有 6 個欄位，每個資訊使用 tab(\t) 分開，換行後為下一組標註資訊



資料處理

Medical Report

Episode No: 09F016547J[IDNum]

091016.NMT[MedicalReport]

SIZAR, HOWARD[PATIENT]

Lab No: 09F01654[IDNum]

.....

Annotation File

10	IDNUM	14	24	09F016547J
10	MEDICALRECORD	25	35	091016.NMT
....				
10	DATE	122	131	24/8/1993 1993-08-24

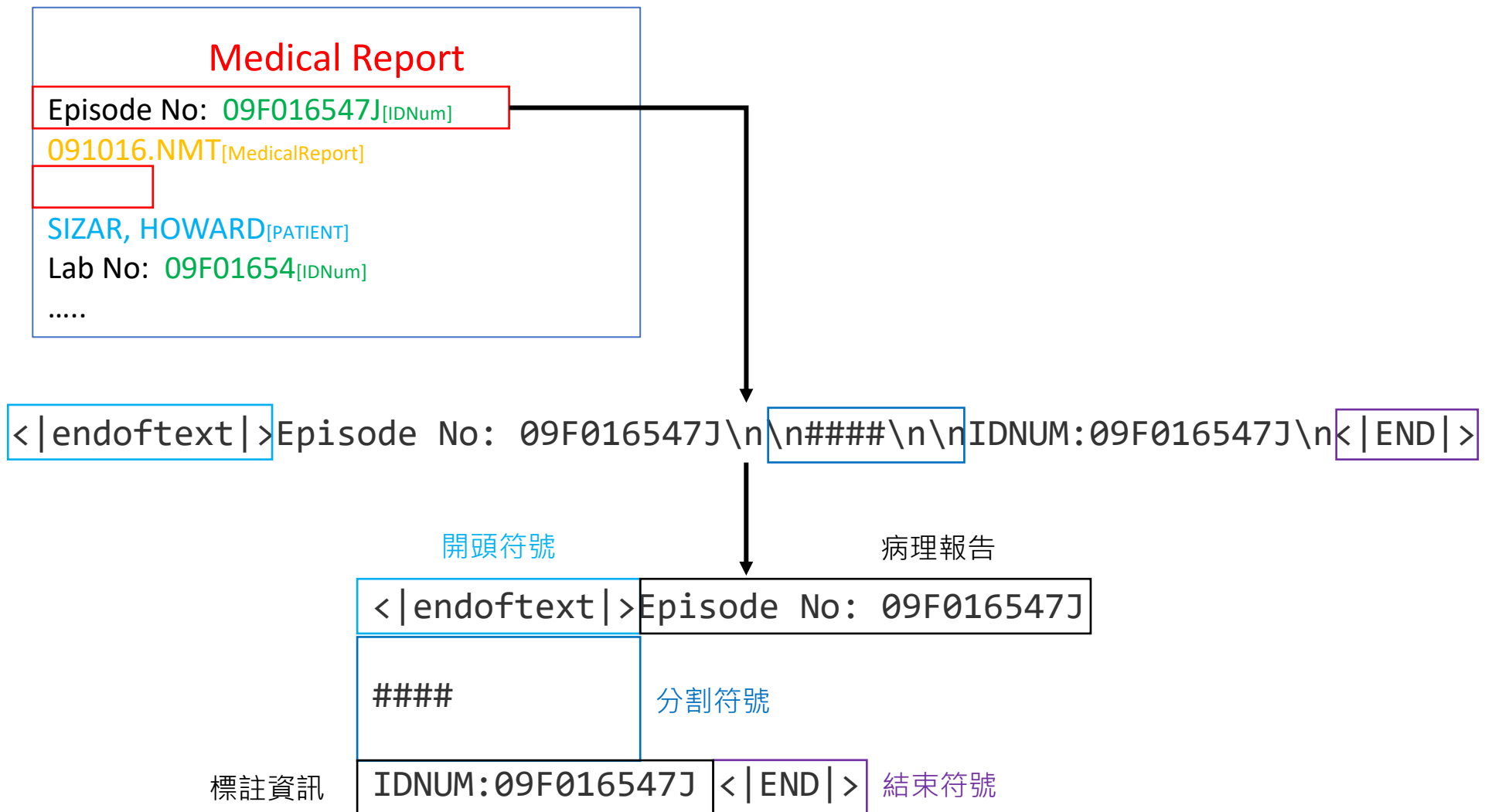
Func **generate_annotated_medical_report**

將每一篇 report 斷句後，
根據 annotation 處理成
sequence pair

<|endoftext|>Episode No: 09F016547J\n\n####\n\nIDNUM:09F016547J\n<|END|>
<|endoftext|>091016.NMT\n\n####\n\nMEDICALRECORD:091016.NMT\n<|END|>
<|endoftext|>\n\n####\n\nPHI:Null<|END|>



資料處理 Cont'd

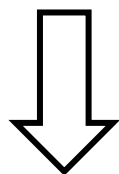




資料處理 Code

```
def generate_annotated_medical_report(anno_file_path ,
                                     titanate):
    anno_lines = read_file(anno_file_path)
    annos_dict = process_annotation_file(anno_lines)
    .....
```

Annotation File				
<u>10</u>	<u>IDNUM</u>	<u>14</u>	<u>24</u>	09F016547J



```
item_dict = {'phi' : "IDNUM",
             'st_idx' : "14",
             'ed_idx' : "24" ,
             'entity' : "09F016547J"}
```

用字典來存取每個檔案的 annotation 資訊

```
def process_annotation_file(lines):
    entity_dict = {}
    for line in lines:
        items = line.strip('\n').split('\t')
        if len(items) == 5:
            item_dict = {
                'phi' : items[1],
                'st_idx' : int(items[2]),
                'ed_idx' : int(items[3]),
                'entity' : items[4],
            }
        elif len(items) == 6:
            item_dict = {
                'phi' : items[1],
                'st_idx' : int(items[2]),
                'ed_idx' : int(items[3]),
                'entity' : items[4],
                'normalize_time' : items[5],
            }
        if items[0] not in entity_dict:
            entity_dict[items[0]] = [item_dict]
        else:
            entity_dict[items[0]].append(item_dict)
    return entity_dict
```




資料處理 Code Cont'd

根據每個檔名來存取檔案全部的 PHI 資訊

```
entity_dict = {"10":  
    [{ 'phi' : "IDNUM",  
      'st_idx' : "14",  
      'ed_idx' : "24",  
      'entity' : "09F016547J"}  
    ,  
    { 'phi' : "IDNUM",  
      'st_idx' : "14",  
      'ed_idx' : "24",  
      'entity' : "09F016547J"}  
    .....]}
```

```
def process_annotation_file(lines):  
    entity_dict = {}  
    for line in lines:  
        items = line.strip('\n').split('\t')  
        if len(items) == 5:  
            item_dict = {  
                'phi' : items[1],  
                'st_idx' : int(items[2]),  
                'ed_idx' : int(items[3]),  
                'entity' : items[4],  
            }  
        elif len(items) == 6:  
            item_dict = {  
                'phi' : items[1],  
                'st_idx' : int(items[2]),  
                'ed_idx' : int(items[3]),  
                'entity' : items[4],  
                'normalize_time' : items[5],  
            }  
        if items[0] not in entity_dict:  
            entity_dict[items[0]] = [item_dict]  
        else:  
            entity_dict[items[0]].append(item_dict)  
    return entity_dict
```



資料處理 Code Cont'd

```
def process_medical_report(txt_name, medical_report_folder, annos_dict, special_tokens_dict):
```

```
    file_name = txt_name + '.txt'
```

```
    sents = read_file(os.path.join(medical_report_folder, file_name))
```

讀取病理報告

```
    article = "".join(sents)
```

```
    bounary , item_idx , temp_seq , seq_pairs = 0 , 0 , "" , []
```

```
    for w_idx, word in enumerate(article):
```

逐個字元迭代

```
        if w_idx == annos_dict[txt_name][item_idx]['st_idx']:
```

如果字元的位置與 annotation 的 PHI 起始位置相同

```
            phi_key = annos_dict[txt_name][item_idx]['phi']
```

```
            phi_value = annos_dict[txt_name][item_idx]['entity']
```

```
            if 'normalize_time' in annos_dict[txt_name][item_idx]:
```

```
                temp_seq += f"{phi_key}:{phi_value}>={annos_dict[txt_name][item_idx]['normalize_time']}\n"
```

```
            else:
```

```
                temp_seq += f"{phi_key}:{phi_value}\n"
```

```
            if item_idx == len(annos_dict[txt_name]) - 1:
```

```
                continue
```

防呆

```
            item_idx += 1
```

```
    if word == '\n':
```

```
        new_line_idx = w_idx + 1
```

```
        if temp_seq == "":
```

```
            temp_seq = "PHI:Null"
```

```
        seq_pair = special_tokens_dict['bos_token'] +
```

```
        article[bounary:new_line_idx] + special_tokens_dict['sep_token'] + temp_seq +
```

```
        special_tokens_dict['eos_token']
```

```
        bounary = new_line_idx
```

```
        seq_pairs.append(seq_pair)
```

```
        temp_seq = ""
```

```
    return seq_pairs
```

取出字典裡對應的資訊
concat 成新的字串

10 IDNUM 14 24 09F016547J



IDNUM:09F016547J\n

10 DATE 122 131 24/8/1993 1993-08-24



DATE : 24/8/1993=>1993-08-24\n



資料處理 Code Cont'd

```
def process_medical_report(txt_name, medical_report_folder, annos_dict, special_tokens_dict):
    file_name = txt_name + '.txt'
    sents = read_file(os.path.join(medical_report_folder, file_name))
    article = "".join(sents)

    bounary , item_idx , temp_seq , seq_pairs = 0 , 0 , "" , []

    for w_idx, word in enumerate(article):
        if w_idx == annos_dict[txt_name][item_idx]['st_idx']:
            phi_key = annos_dict[txt_name][item_idx]['phi']
            phi_value = annos_dict[txt_name][item_idx]['entity']
            if 'normalize_time' in annos_dict[txt_name][item_idx]:
                temp_seq += f"{phi_key}:{phi_value}>={annos_dict[txt_name][item_idx]['normalize_time']}\n"
            else:
                temp_seq += f"{phi_key}:{phi_value}\n"
            if item_idx == len(annos_dict[txt_name]) - 1:
                continue
            item_idx += 1
```

if word == '\n': 以\n作為斷句符號

new_line_idx = w_idx + 1 記錄下個句子開始的位置

if temp_seq == "":
temp_seq = "PHI:Null" 如果該句子都沒有 PHI 資訊再裡面則為 Null

seq_pair = special_tokens_dict['bos_token'] +
article[bounary:new_line_idx] + special_tokens_dict['sep_token'] + temp_seq +
special_tokens_dict['eos_token'] 開頭符號+句子+分割符號+PHI 資訊+結束符號

bounary = new_line_idx 新的位置

seq_pairs.append(seq_pair)

temp_seq = ""

return seq_pairs



資料處理 Code Cont'd

Episode No: 09F016547J

...

REMARK TAS 5084

Specimen: Tissue

...

Collected: 28/08/2013 at 08:26

<|endoftext|>Episode No: 09F016547J

####

IDNUM:09F016547J

<|END|>

<|endoftext|>REMARK TAS 5084

####

CITY:REMARK

STATE:TAS

ZIP:5084

<|END|>

<|endoftext|>Collected: 28/08/2013 at 08:26

####

TIME:28/08/2013 at 08:26=>2013-08-28T08:26

<|END|>



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

- 病患隱私資訊擷取子任務
- 時間資訊正規化子任務

批次資料轉換

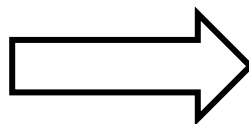


TorchDataset

Text

```
<|endoftext|>Episode No: 09F016547J  
####  
IDNUM:09F016547J  
<|END|>
```

斷詞



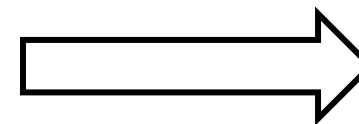
Token

```
<|endoftext|>  
Ep  
isode  
GNo  
:  
\s  
09  
F  
01  
65  
47  
J  
C  
\n####\n\nID  
NUM  
:  
09  
F  
01  
65  
47  
J  
C  
<|END|>
```

Token id

```
0  
22098  
22151  
1621  
27  
50276  
2693  
39  
520  
2082  
2504  
43  
187  
50279  
1838  
20872  
27  
2693  
39  
520  
2082  
2504  
43  
187  
50277
```

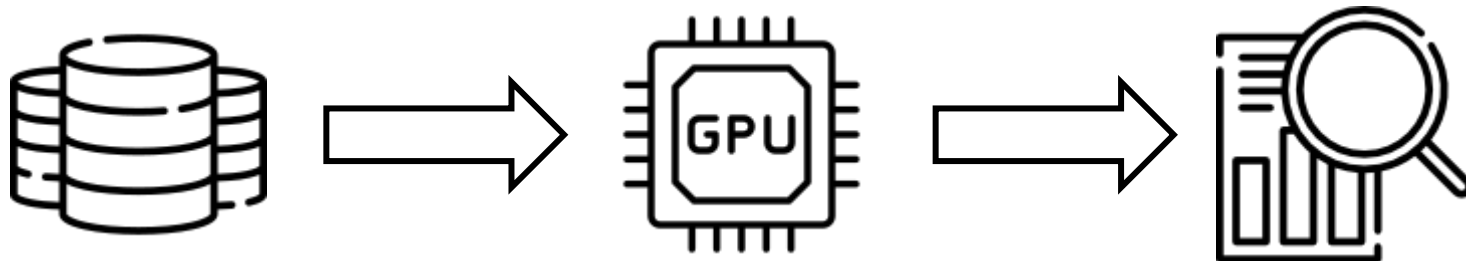
每個token的index



轉換成機器看得懂的语言



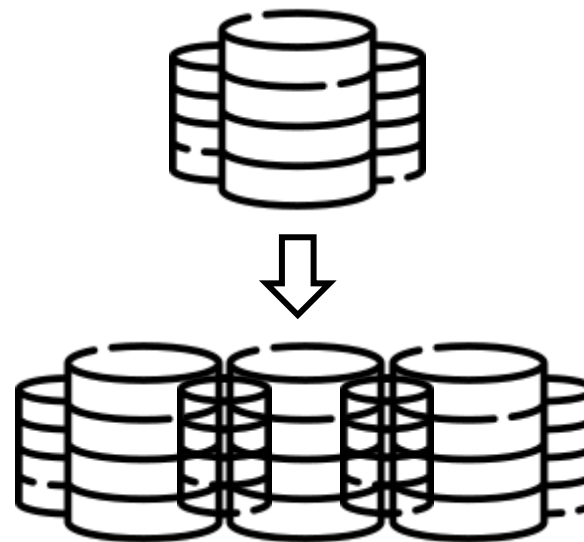
DataLoader Usage



GPU處理資料效率好

- 並行處理能力
- 高內存頻寬
- 向量化操作
- CUDA和類似的平行計算框架
- 大規模且高效的矩陣操作

要先把資料處理成批次資料





Pytorch Parallel Dataset



轉換成 torch 格式的 data
方便 Dataloader 呼叫

轉換為批次的 torch data



TorchDataset

class GPTDataset(Dataset):

```
def __init__(self, seq_paris, tokenizer , special_tokens_dict , pad_idx):
    self.seq_paris = seq_paris
    self.tokenizer = tokenizer
    self.special_tokens_dict = special_tokens_dict
    self.pad_idx = pad_idx
    self.mode = mode
```

初始化、自訂義變數

```
def __len__(self):
    return len(self.seq_paris)
```

檢視全部資料長度

```
def __getitem__(self, index):
    return self.seq_paris[index]
```

呼叫特定索引的資料

```
def collate_batch(self , datasets):
```

```
    tokens_list , labels_list , attention_mask_list = [] , [] , []
    for dataset in datasets:
        encoded_seq = tokenizer(dataset)
        indexed_tks = encoded_seq["input_ids"]
        attention_mask = encoded_seq["attention_mask"]
```

```
        tokens_list.append(torch.tensor(indexed_tks))
        labels_list.append(torch.tensor(indexed_tks))
        attention_mask_list.append(torch.tensor(attention_mask))
    return self.pad_sequence(tokens_list , labels_list , attention_mask_list)
```

繼承 TorchDataset 的函式

處理批次資料的函式



TorchDataset Cont'd

一組批次資料

```
def collate_batch(self , datasets):
```

```
    tokens_list , labels_list , attention_mask_list = [] , [] , []
```

```
    for dataset in datasets:
```

```
        encoded_seq = tokenizer(dataset)
```

```
        indexed_tks = encoded_seq["input_ids"]
```

轉換字串

```
        attention_mask = encoded_seq["attention_mask"]
```

模型著重注意的資訊

```
        tokens_list.append(torch.tensor(indexed_tks))
```

```
        labels_list.append(torch.tensor(indexed_tks))
```

```
        attention_mask_list.append(torch.tensor(attention_mask))
```

```
    return self.pad_sequence(tokens_list , labels_list , attention_mask_list)
```

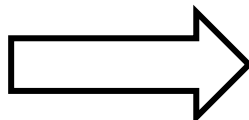
<|endoftext|>Episode No: 09F016547J

####

IDNUM:09F016547J

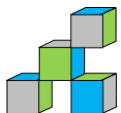
<|END|>

Text



[0, 22098, 22151, 1621, 27, 50276,
2693, 39, 520, 2082, 2504, 43, 187,
50279, 1838, 20872, 27, 2693, 39,
520, 2082, 2504, 43, 187, 50277]

Token id



TorchDataset Code

```
def pad_sequence(self, non_pad_token, non_pad_label, non_pad_attn):
    max_size = max([len(ele) for ele in non_pad_token])
```

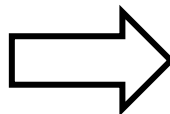
找出該批次資料中的最長序列的長度

```
    pad_batch1 = torch.stack([torch.cat([t, torch.LongTensor([self.pad_idx] * (max_size - len(t)))] for
t in non_pad_token)])
    pad_batch2 = torch.stack([torch.cat([t, torch.LongTensor([self.pad_idx] * (max_size - len(t)))] for
t in non_pad_label)])
    pad_batch3 = torch.stack([torch.cat([t, torch.LongTensor([0] * (max_size - len(t)))] for t in
non_pad_attn)])
    return pad_batch1, pad_batch2, pad_batch3
```

把每個批次序列展開成一樣的長度

Batch_size = 5

[1, 5, 3, 5, 8, 2, 7]
 [2, 6, 7]
 [4, 2, 9, 2]
 [5, 3, 1, 7, 8]
 [2, 6, 4, 9]



Max_size = 7
 Pad_idx = 0

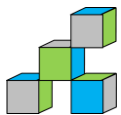
[1, 5, 3, 5, 8, 2, 7]
 [2, 6, 7, 0, 0, 0, 0]
 [4, 2, 9, 2, 0, 0, 0]
 [5, 3, 1, 7, 8, 0, 0]
 [2, 6, 4, 9, 0, 0, 0]

Padding Token

讓模型關注 1 的部分
 可以讓訓練更有效率

[1, 1, 1, 1, 1, 1, 1]
 [1, 1, 1, 0, 0, 0, 0]
 [1, 1, 1, 1, 0, 0, 0]
 [1, 1, 1, 1, 1, 0, 0]
 [1, 1, 1, 1, 0, 0, 0]

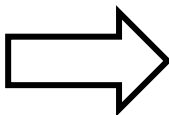
Padding attention mask



TorchDataset Code

Batch_size = 5

[1, 5, 3, 5, 8, 2, 7]
[2, 6, 7]
[4, 2, 9, 2]
[5, 3, 1, 7, 8]
[2, 6, 4, 9]



[1, 5, 3, 5, 8, 2, 7]
[2, 6, 7, 0, 0, 0, 0]
[4, 2, 9, 2, 0, 0, 0]
[5, 3, 1, 7, 8, 0, 0]
[2, 6, 4, 9, 0, 0, 0]

Padding Token

[1, 1, 1, 1, 1, 1, 1]
[1, 1, 1, 0, 0, 0, 0]
[1, 1, 1, 1, 0, 0, 0]
[1, 1, 1, 1, 1, 0, 0]
[1, 1, 1, 1, 0, 0, 0]

Padding attention mask



[1, 5, 3, 5, 8, 2, 7]
[2, 6, 7, 0, 0, 0, 0]
[4, 2, 9, 2, 0, 0, 0]
[5, 3, 1, 7, 8, 0, 0]
[2, 6, 4, 9, 0, 0, 0]

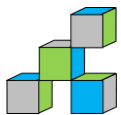
Padding Token



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

- 病患隱私資訊擷取子任務
- 時間資訊正規化子任務

模型訓練



TorchDataset & Model

```
BATCH_SIZE = 16
```

```
plm = "EleutherAI/pythia-70m"
```

語言模型與GPT很相似
參數較少的版本

```
tokenizer = AutoTokenizer.from_pretrained(plm)
```

```
tokenizer.add_special_tokens(special_tokens_dict)
```

```
PAD_IDX = tokenizer.convert_tokens_to_ids(tokenizer.pad_token)
```

```
tr_dataset = GPTDataset(train_seq_pairs,  
                        tokenizer,  
                        special_tokens_dict,  
                        PAD_IDX)
```

```
bucket_train_dataloader = DataLoader(tr_dataset,
```

```
    batch_size=BATCH_SIZE,
```

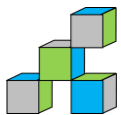
Dataloader 迭代時呼叫的函式

```
    collate_fn=tr_dataset.collate_batch)
```

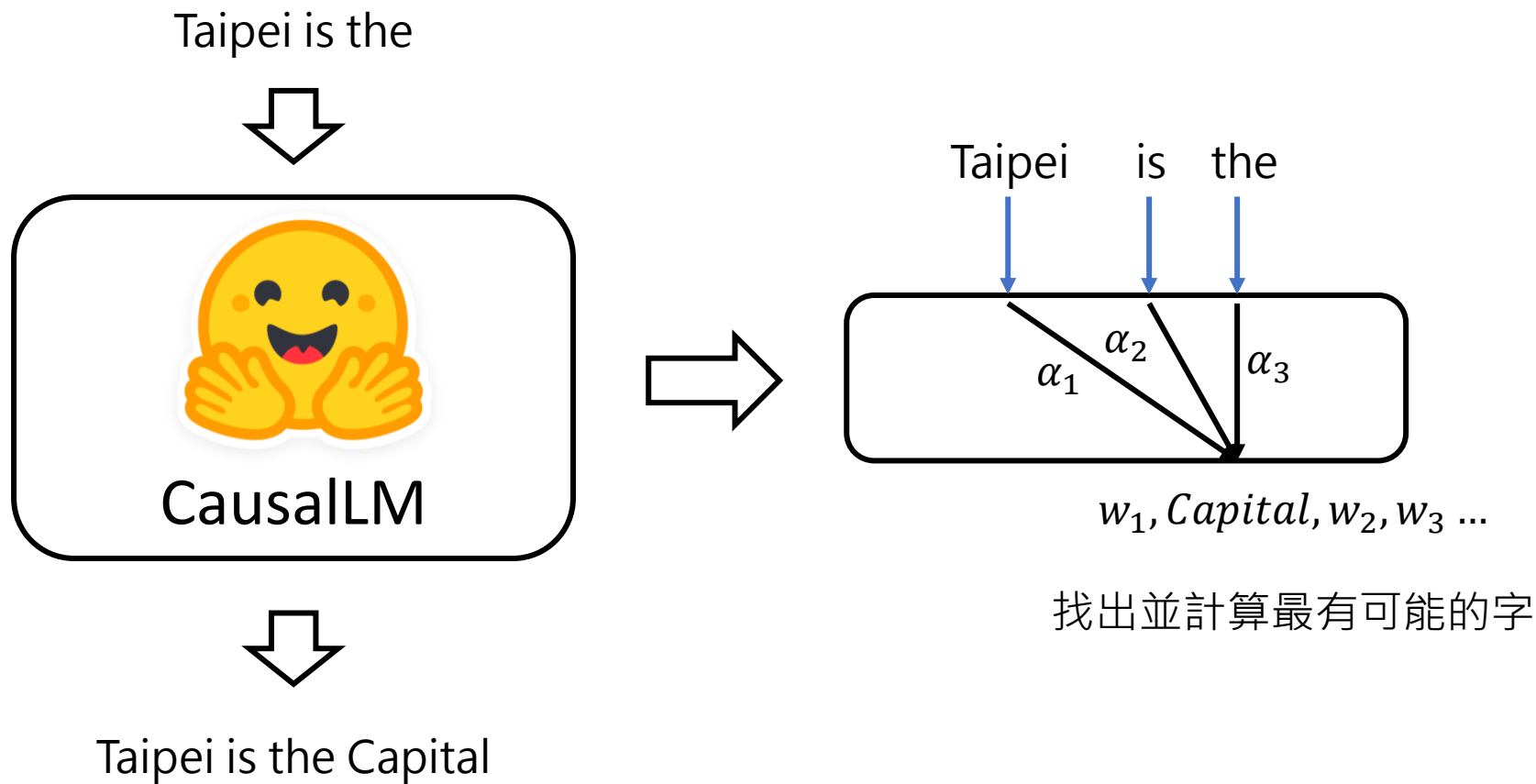
```
model = AutoModelForCausalLM.from_pretrained(plm)
```

```
model.resize_token_embeddings(len(tokenizer))
```

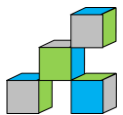
模型宣告



Causal Language Model



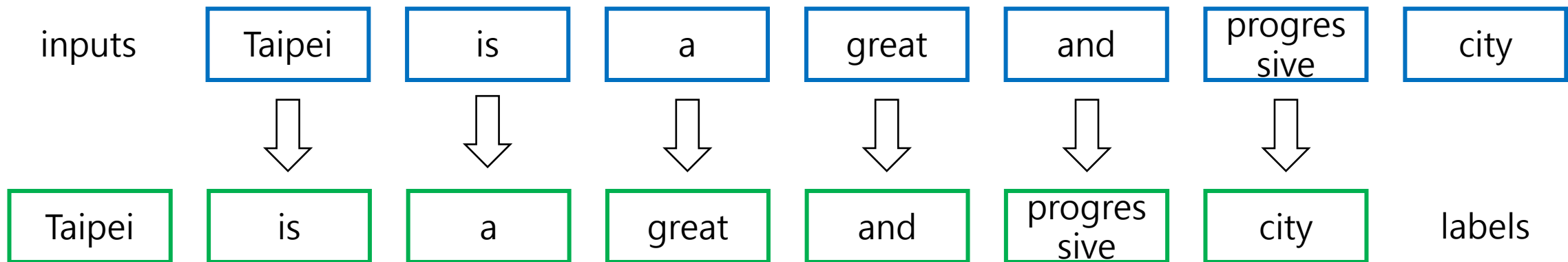
GPT 系列生成式模型的訓練方式



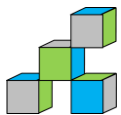
Causal Language Model Cont'd

Taipei is the <blank> .

CLM 會學著去預測下一個 token



Input 與 label 是一樣的序列

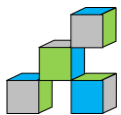


EleutherAI/pythia-70m



current Pythia suffix	old suffix	total params	non-embedding params
70M	19M	70,426,624	18,915,328
160M	125M	162,322,944	85,056,000
410M	350M	405,334,016	302,311,424
1B	800M	1,011,781,632	805,736,448
1.4B	1.3B	1,414,647,808	1,208,602,624
2.8B	2.7B	2,775,208,960	2,517,652,480
6.9B	6.7B	6,857,302,016	6,444,163,072
12B	13B	11,846,072,320	11,327,027,200
GPT2		1.5B	
GPT3		1750B	

不同參數量大小



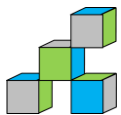
HyperParameters

優化器 & 優化器的參數調整

```
param_optimizer = list(model.named_parameters())
no_decay = ['bias', 'LayerNorm.weight']
optimizer_grouped_parameters = [
    {'params': [p for n, p in param_optimizer if not any(nd in n for nd in no_decay)],
      'weight_decay_rate': 0.01},
    {'params': [p for n, p in param_optimizer if any(nd in n for nd in no_decay)],
      'weight_decay_rate': 0.01}
]
optimizer = AdamW(
    optimizer_grouped_parameters,
    lr=3e-5
)
```

```
epochs = 20
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
```

迭代次數 & GPU 設定



Model Training

```
for _ in trange(epochs, desc="Epoch"):
    model.train()
```

```
for step, (seqs, labels, masks) in enumerate(bucket_train_dataloader):
```

```
    seqs = seqs.to(device)
    labels = labels.to(device)
    masks = masks.to(device)
```

要進模型的資料丟進 GPU

```
    model.zero_grad()
```

模型向前傳播

```
    outputs = model(seqs, labels=labels, attention_mask=masks)
```

```
    logits = outputs.logits
```

```
    loss = outputs.loss
```

```
    loss = loss.mean()
```

```
    total_loss += loss.item()
```

```
    loss.backward()
```

向後傳播計算 Loss

```
    optimizer.step()
```

更新梯度



Decoding

生成 Token 數量限制

```
def sample_text(model, tokenizer, text, n_words=100):
    model.eval()
    text = tokenizer.encode(text)
    inputs, past_key_values = torch.tensor([text]).to(device), None

    with torch.no_grad():
        for _ in range(n_words):
            out = model(inputs, past_key_values=past_key_values)
            logits = out.logits
            past_key_values = out.past_key_values
            log_probs = F.softmax(logits[:, -1], dim=-1)
            inputs = torch.multinomial(log_probs, 1)
            text.append(inputs.item())
            if tokenizer.decode(inputs.item()) == eos:
                break

    return tokenizer.decode(text)
```

文字接龍的形式



Decoding Cont'd

生成 Token 數量限制

```
def sample_text(model, tokenizer, text, n_words=100):
    model.eval()
    text = tokenizer.encode(text)
    inputs, past_key_values = torch.tensor([text]).to(device), None

    with torch.no_grad():
        for _ in range(n_words):
            out = model(inputs, past_key_values=past_key_values)
            logits = out.logits
            past_key_values = out.past_key_values
            log_probs = F.softmax(logits[:, -1], dim=-1)
            inputs = torch.multinomial(log_probs, 1)
            text.append(inputs.item())
            if tokenizer.decode(inputs.item()) == eos:
                break

    return tokenizer.decode(text)
```

文字接龍的形式



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

- 病患隱私資訊擷取子任務
- 時間資訊正規化子任務

上傳檔案注意事項



上傳檔案注意事項

```
838 ZIP 101 105 3144
838 DATE 131 140 27/5/2036 2036-05-27
838 DEPARTMENT 191 208 DICKINSON 3 NORTH
838 HOSPITAL 209 227 PETER JAMES CENTRE
838 DATE 325 332 18/9/63 2063-09-18
838 DATE 1913 1920 12.9.63 2063-09-12
838 DATE 3518 3525 18/9/63 2063-09-18
838 DATE 3692 3699 24.9.63 2063-09-24
```

非時間資訊要有 5 個欄位：檔案名稱\t隱私欄位\t起始位置\t結束位置\t隱私資訊

時間資訊要有 6 個欄位：檔案名稱\t隱私欄位\t起始位置\t結束位置\t隱私資訊\t時間正規化欄位



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

- 病患隱私資訊擷取子任務
- 時間資訊正規化子任務

結論與未來方向

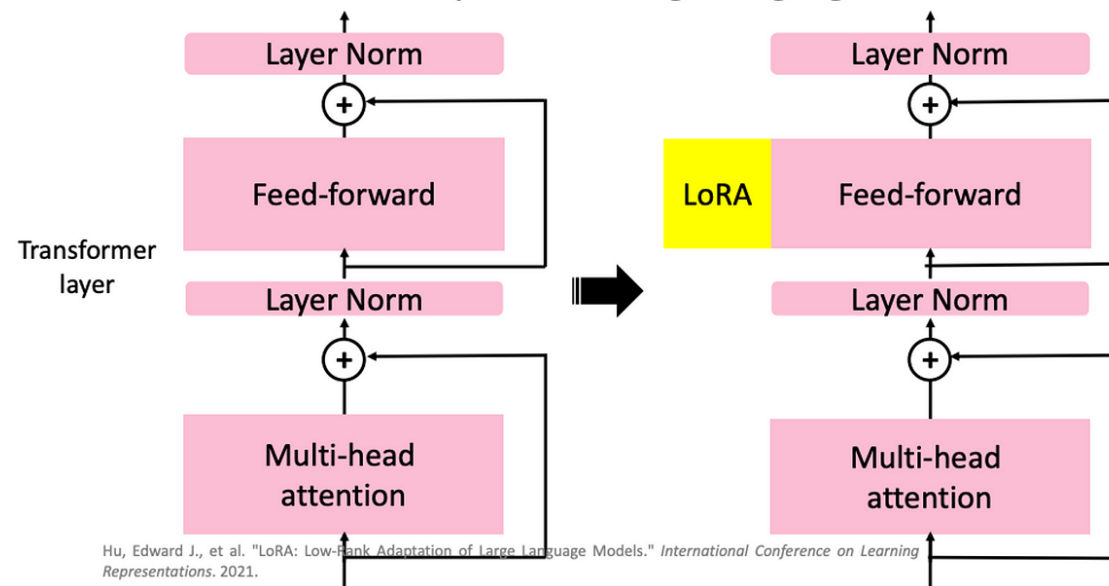


Enhance Tips

- 調整 HyperParameters
ex. epochs, learning rate, optimizer.....
- 訓練時加入 scheduler
- pythia 模型參數量提高 (note. GPU Memory)
- 使用 GPT2、GPTJ (note. 搭配 adapter)

current Pythia suffix	old suffix	total params
70M	19M	70,426,624
160M	125M	162,322,944
410M	350M	405,334,016
1B	800M	1,011,781,632
1.4B	1.3B	1,414,647,808
2.8B	2.7B	2,775,208,960
6.9B	6.7B	6,857,302,016
12B	13B	11,846,072,320
GPT2		1.5B
GPT3		1750B

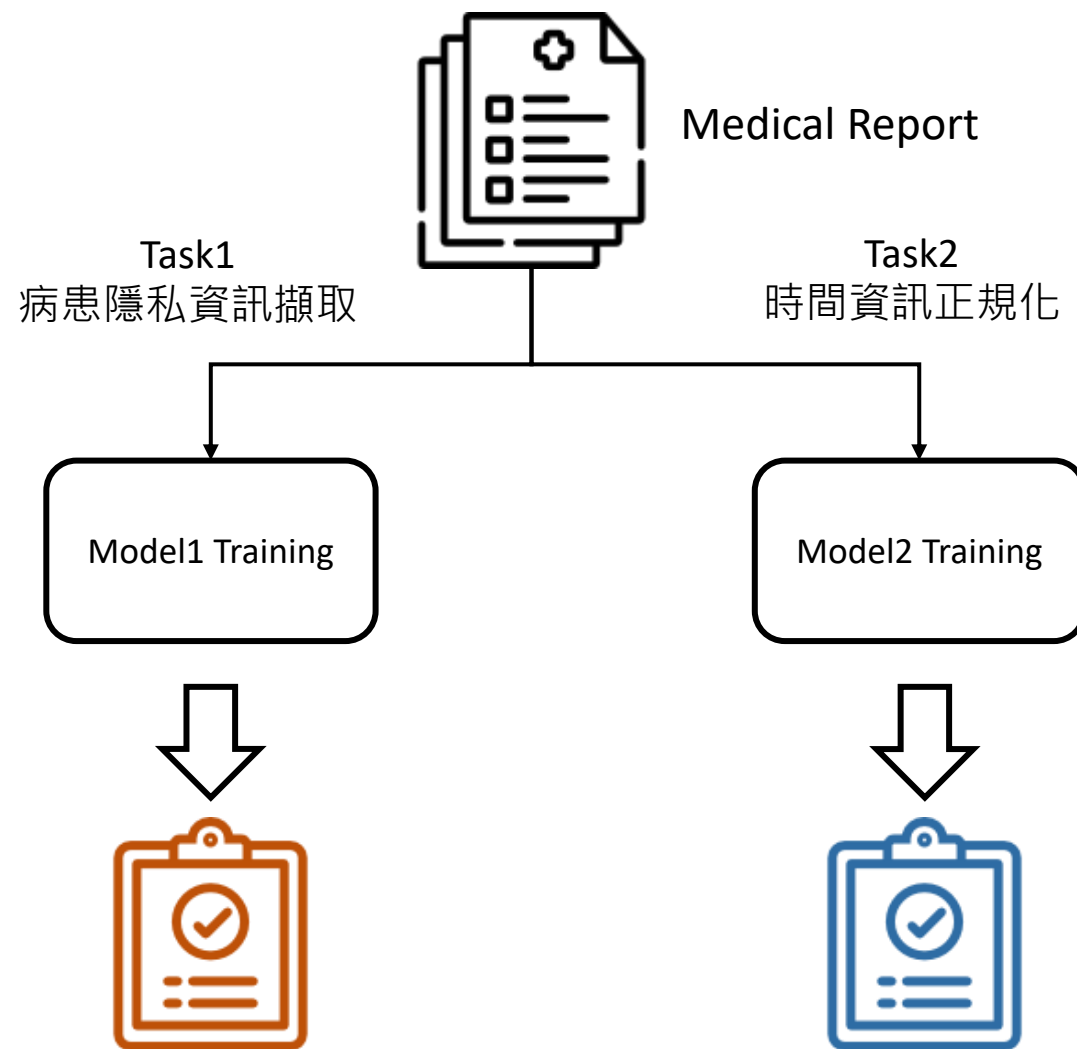
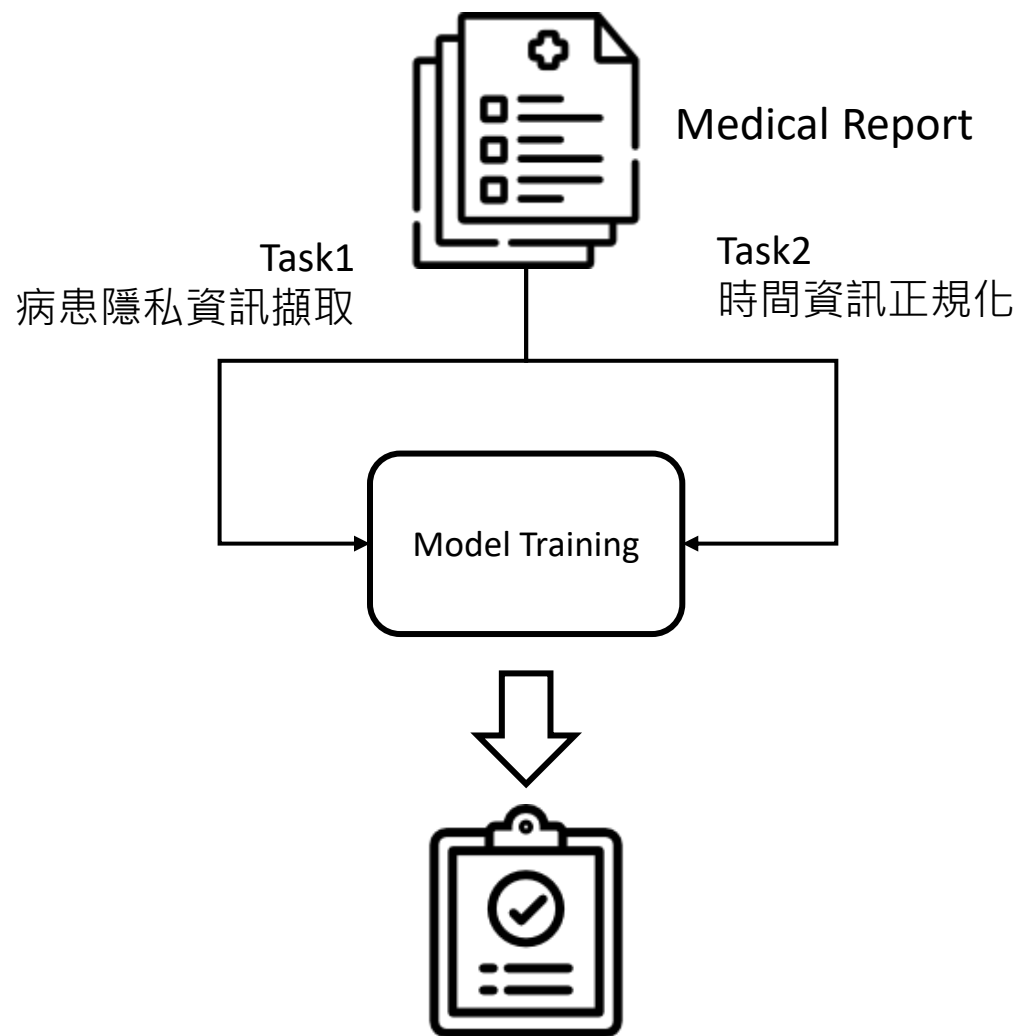
• LoRA: Low-Rank Adaptation of Large Language Models



Hu, Edward J., et al. "LoRA: Low-Rank Adaptation of Large Language Models." *International Conference on Learning Representations*. 2021.



Enhance Tips Cont'd



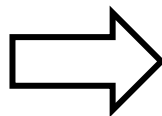


Enhance Tips Cont'd

Medical Report

Episode No: 09F016547J[IDNum]
091016.NMT[MedicalReport]

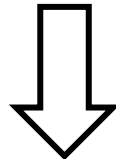
SIZAR, HOWARD[PATIENT]
Lab No: 09F01654[IDNum]
.....



<|endoftext|>Episode No: 09F016547J
091016.NMT

####

IDNUM:09F016547J
MEDICALRECORD:091016.NMT<|END|>



<|endoftext|>Episode No:
09F016547J\n\n####\n\nIDNUM:09F016547J\n<|END|>

<|endoftext|>091016.NMT\n\n####\n\nMEDICALRECOR
D:091016.NMT\n<|END|>

<|endoftext|>\n\n####\n\nPHI:Null<|END|>

將原本單靠\n換行
現在讓更多行變成一個句子
讓模型更好學習上下文

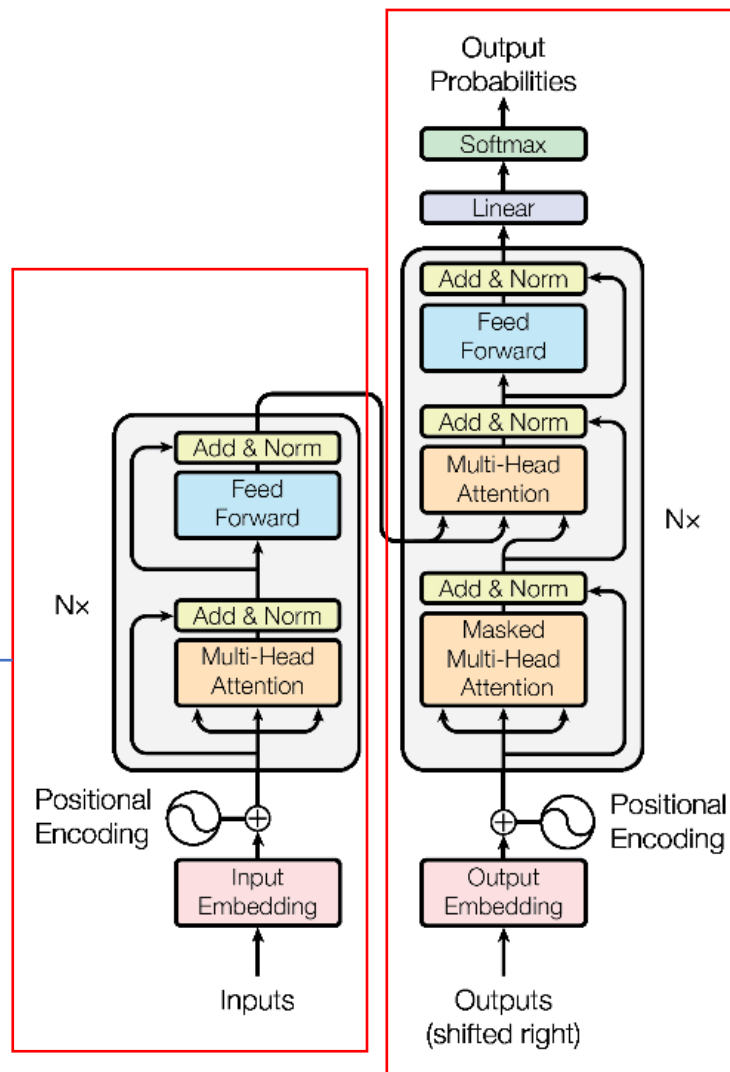
※ 注意句子長度



Enhance Tips Cont'd



BERT 2018~



GPT



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

- 病患隱私資訊擷取子任務
- 時間資訊正規化子任務

[程式相關連結](#)

END