



Project2 b. Report

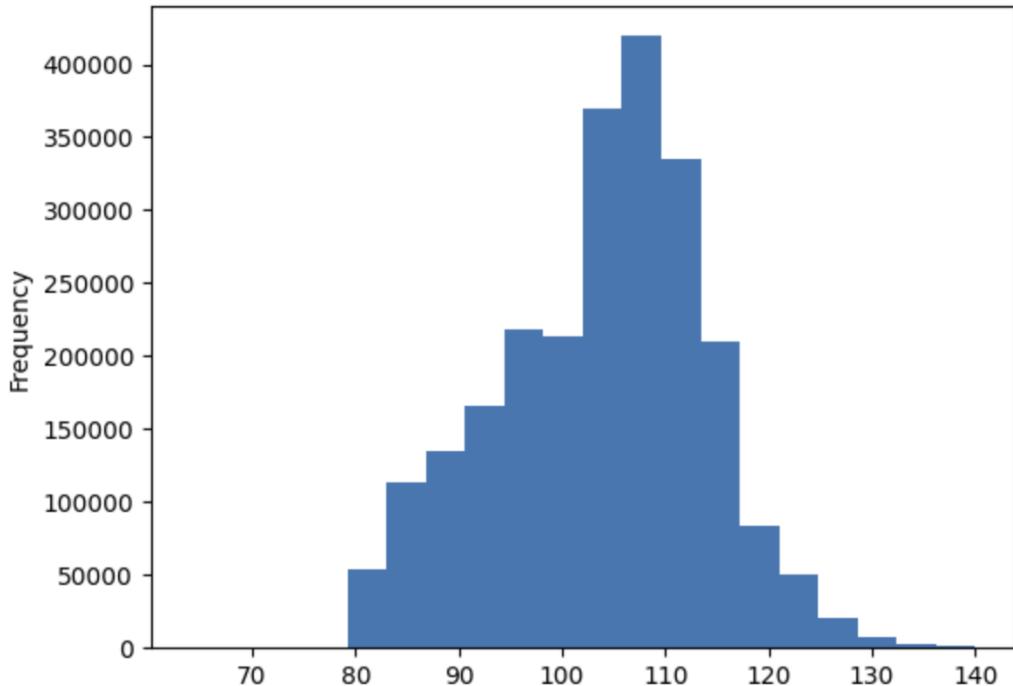
| 資訊系大四 F74092269 陳冠廷

任務介紹

作業 b 為簡體中文的文本摘要任務，透過微調預訓練的語言模型（T5 與 GPT-2）來實現文本摘要的技術。

資料集

- LCSTS 中文摘要資料集
- 約有 2.4M 筆訓練資料和 8.7K 筆驗證資料
- 資料集長度分布 (x 軸為文章字數)



- 由於資料集巨大但是運算資源有限，我只篩選約莫1%的原訓練資料作為訓練資料。而驗證資料則不做採樣。同時考慮到不同長度的文本或多或少會影響到模型的表現，因此我簡單將資料分為三種長度區間，個別採樣相同數量，如下程式碼：

```

def group_length(length):
    # 依據長度切分三類
    if length <= 95:
        return 0
    elif 95 < length <= 115:
        return 1
    else:
        return 2

class SummaryDataset(Dataset):
    def __init__(self, split="train", ratio=1.0) -> None:
        super().__init__()
        assert split in ["train", "validation", "test"]

        # 下載資料
        df = load_dataset("hugcyp/LCSTS", split=split, cache_dir="./cache/")
        # 建立 text_length 長度欄位
        df["text_length_group"] = df["text"].apply(lambda text: group_length(text))

        if (ratio < 1.0): # 使用隨機採樣
            # 每種長度區間採樣相同數量
            n_sample = int(len(df) * ratio / 3)
            self.data_df = df.groupby("text_length_group").sample(n=n_sample,
        else:
            # 不採樣
            self.data_df = df
    
```

```

def __getitem__(self, index):
    return self.data_df.loc[index, :]

def __len__(self):
    return len(self.data_df)

```

- 由於 T5 類的模型為encoder-decoder架構，因此資料集非常容易包裝，只需將**文本設為輸入資料而摘要設為預測目標**即可

```

def get_tensor(sample, tokenizer):
    # 將模型的輸入和ground truth打包成Tensor
    model_inputs = tokenizer.batch_encode_plus([each["text"] for each in sample])
    model_outputs = tokenizer.batch_encode_plus([each["summary"] for each in sample])
    return model_inputs["input_ids"].to(device), model_outputs["input_ids"].to(device)

```

- 對於 gpt-2 這類decoder-based的模型就不能簡單拆分資料，而是要考慮到文字接龍的過程來設計資料形式，即輸出就是輸入向右位移一位，而且**將文本與摘要放在一起成為一段文字當作輸入**。另外在 GPT-2 模型中會自己右移資料，因此只需準備相同的input和output，但是為了避免模型學習到錯誤的padding知識，因此會**將outputs在padding的地方設為-100**，而模型就會自動忽略其對損失函數的影響。我特別標註了**最後一個padding必須將-100重設為結尾符號**，讓模型學習到甚麼時候該停止。

```

def get_gpt_tensor(sample, tokenizer, mask_non_summary=False):
    # 將模型的輸入和ground truth打包成Tensor
    model_inputs = tokenizer([each["text"] + " " + tokenizer.eos_token + " " for each in sample])
    inputs = model_inputs["input_ids"]
    outputs = inputs.clone()

    # 避免 padding 的地方算到loss
    outputs[outputs == tokenizer.pad_token_id] = -100
    # 終止要學習
    outputs[:, -1] = tokenizer.pad_token_id

    return inputs.to(device), outputs.to(device)

```

模型選擇

- google/mT5-base
 - 多語言的T5 model，可以編碼中文
 - encoder-decoder架構**：透過encoder有效編碼前文，在綜合前文的情況下，生成特定任務的後文。例如：此次任務的前文為原文，後文為摘要。encoder先編碼文章內容後，再根據任務所需生成摘要，並指比對摘要與生成內容的差異來優化參數

A 範例程式的 Flan-T5 並不能讀中文，進行實作時發現中文皆被tokenizer編碼為unknown，因此一開始時一直碰壁

- openai/gpt-2

- 不同於Flan-T5，GPT-2使用特殊的編碼方法，因此可以直接編碼中文
- **decoder架構**：根據前文去預測下一個字，屬於文字接龍的方法

訓練方法

- google/mT5-base

- 設定以下超參數

參數名	參數值
learning_rate	5e-5
epochs	3
batch_size	16
optimizer	AdamW
lr_scheduler	CosineAnnealingLR
step	300

- 訓練過程一般的學習方法，並設定step來評價模型在驗證資料上的loss，用這個loss值來決定模型有沒有比較好。當模型的驗證損失比目前的最佳驗證損失小時，我才會去計算實際的rouge分數，並存下此模型的權重

```

def train(model, train_loader, valid_loader, model_name):

    # 紀錄訓練狀況
    train_losses = []
    eval_losses = []
    total_loss, total_count, curr_step, best_loss = 0, 0, 0, 10000

    optimizer = AdamW(model.parameters(), lr = learning_rate)
    scheduler = lr_scheduler.CosineAnnealingLR(
        optimizer=optimizer,
        T_max=epochs * len(train_loader.dataset)
    )

    for ep in range(epochs):

        pbar = tqdm(train_loader)
        pbar.set_description(f"Training epoch [{ep+1}/{epochs}]")

        for inputs, targets in pbar:
            # 清空 gradient
            optimizer.zero_grad()
            # 更新參數
            loss = model(input_ids=inputs, labels=targets).loss
            loss.backward()
            optimizer.step()

            # 更新訓練狀況
            total_loss += loss.item()
    
```

```

        total_count += inputs.shape[0]
        curr_step += 1
        pbar.set_postfix(loss = round(total_loss/total_count, 4))

    scheduler.step()

    # 達到驗證step數
    if curr_step % step == 0:
        train_losses.append(round(total_loss/total_count, 4))
        eval_loss = get_eval_loss(model, valid_loader)
        eval_losses.append(eval_loss)

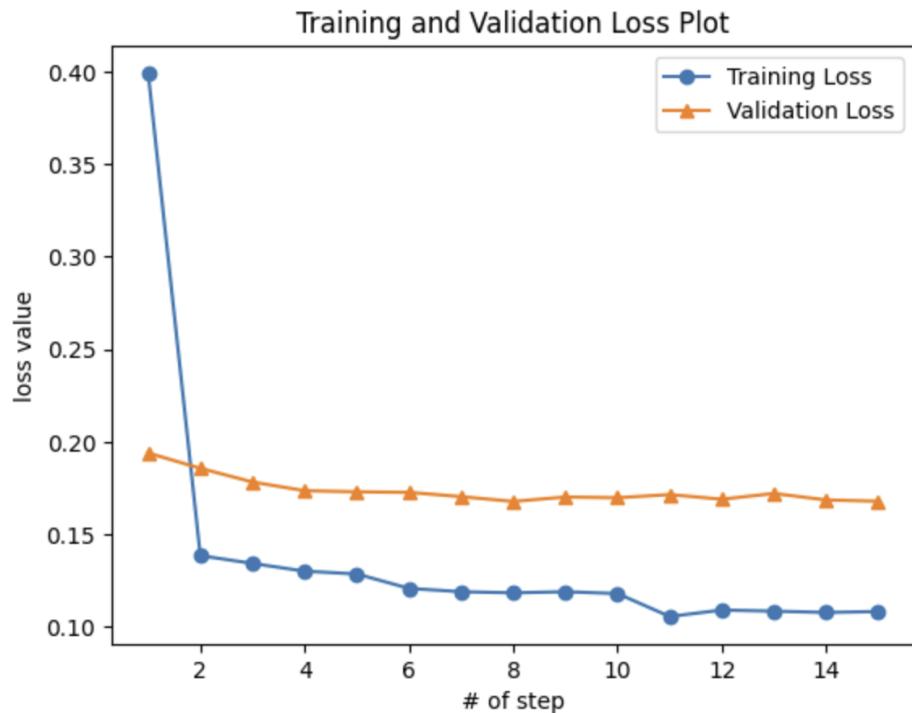
        total_loss = 0
        total_count = 0

        if eval_loss < best_loss:
            best_loss = eval_loss
            # 驗證模型分數並儲存
            torch.save(model, f'{model_name}_best.mod')
            print(f"Rouge-2 score on step {curr_step // step}:",
                  eval_loss)

    return train_losses, eval_losses

```

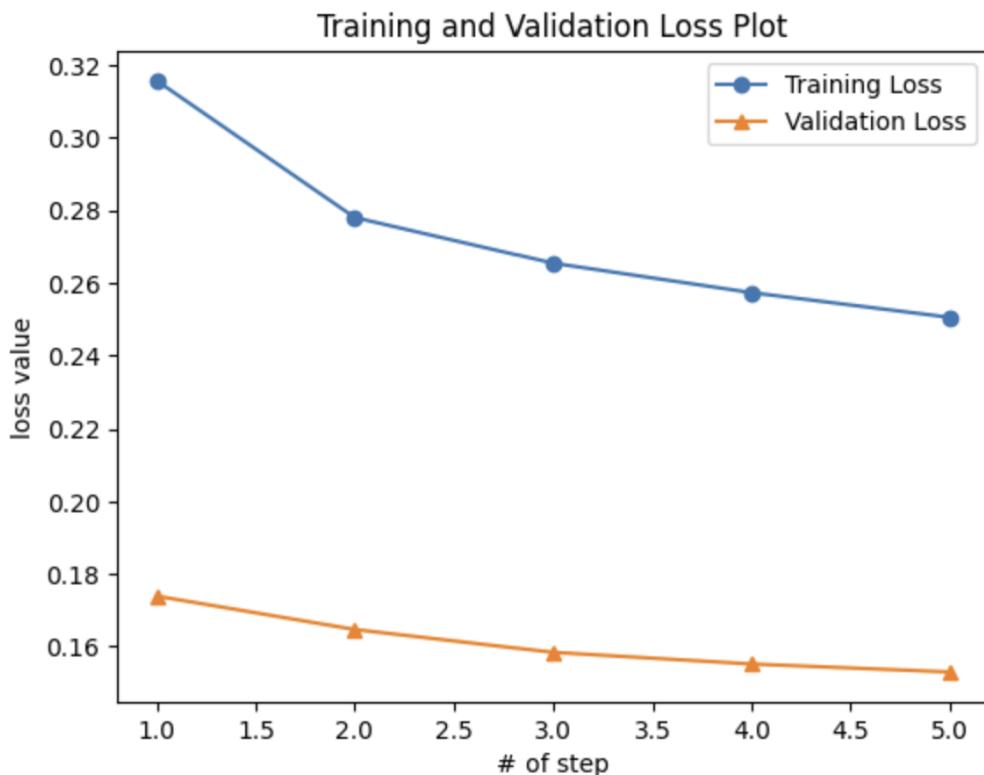
- Loss 圖 (在step個iteration中的平均損失走勢圖)



- openai/gpt-2
 - 設定以下超參數，由於訓練時間長且驗證時間更長，只執行了1個epoch

參數名	參數值
learning_rate	3e-4
epochs	1
batch_size	8
optimizer	AdamW
lr_scheduler	CosineAnnealingLR
step	600

- 訓練過程一般的學習方法，並設定step來評價模型在驗證資料上的loss，用這個loss值來決定模型有沒有比較好。當模型的驗證損失比目前的最佳驗證損失小時，我才會去計算實際的rouge分數，並存下此模型的權重（同於mT5的程式碼）
- Loss 圖 (在step個iteration中的平均損失走勢圖)



評分指標

- 我選擇rouge-L和rouge-2作為我的評價指標，主要用來評斷關鍵字詞的比對狀況，前者考慮的是最大子序列，後者則是2-gram。
- mT5
 - evaluation相對來說也簡單，基本的清除掉結果的<pad>和</s>這些不是我們關注的文字內容後，就可以算rouge了。比較值得注意的是，原本sample code是用英文來算分，因此會以空格來切割文字，但是中文段字應該是直接一個一個字切割，直接使用list函數即可。

```
def evaluate(model, valid_loader):
```

```

print("Start evaluation...")

for inputs, targets in valid_loader:

    outputs = [each.replace("<pad>", "").replace("</s>", "") for each
    targets = [each.replace("<pad>", "").replace("</s>", "") for each

        for i in range(len(outputs)):
            # 中文不是用空格當分割 -> 而是用一個一個字 -> list 切開 string
            candidate = list(outputs[i])
            ground_truth = [list(targets[i])]
            rouge.update(([candidate], [ground_truth]))


    return rouge.compute()

```

- Finetune 過程中我所擷取到的最佳mT5表現

Rouge-L-P	Rouge-L-R	Rouge-L-F	Rouge-2-P	Rouge-2-R	Rouge-2-F
0.313	0.246	0.246	0.213	0.164	0.164

- 範例生成結果：有些內容確實是有摘要出重點，甚至換句話說（4, 5, 6, 7），但有些內容卻抓錯重點而生成錯誤主題（1, 3, 8），在沒有細緻地調整參數和使用少部分資料而已，已能做到不錯的成果。

```

===== 1 =====
---> Ground Truth: 林志颖公司疑涉虚假营销无厂房无研发
---> Prediction : 爱碧丽假保健品售价高达4元
===== 2 =====
---> Ground Truth: 从韩亚航空事故看其应对路径
---> Prediction : 三位一体呵护韩国形象
===== 3 =====
---> Ground Truth: 女子用板车拉九旬老母环游中国1年走2万4千里
---> Prediction : 妈妈“一辈子在锅台边转”
===== 4 =====
---> Ground Truth: 银行集体发声:房贷政策没变
---> Prediction : 多家银行:房价不会大涨大跌
===== 5 =====
---> Ground Truth: 四律师上书民航总局:起飞前应公布机长信息
---> Prediction : 民航局:航班起飞前要向乘客公布机组人员信息
===== 6 =====
---> Ground Truth: 钱理群“告别教育”
---> Prediction : 钱理群:教育无立足之地
===== 7 =====
---> Ground Truth: 中国游客大增多国放宽签证
---> Prediction : 中泰免签政策
===== 8 =====
---> Ground Truth: 信披违规外加业绩亏损*ST国创退市风险概率大增
---> Prediction : 沪市公司被立案稽查""黑名单"""

```

- GPT-2

- evaluation較困難，需要切出哪些部分為prompt哪些為答案。

```

def evaluate(model, valid_loader):

    for inputs, targets in tqdm(valid_loader, leave=True):

        indices = (inputs==tokenizer.pad_token_id).nonzero()
        prompts = []
        answers = []

        for n_batch in range(len(inputs)):
            # 找出所有 padding 位置，並針對倒數三個 -> left padding 結尾、target_start
            pos_pad = indices[ indices[:, 0] == n_batch ]
            if len(pos_pad) > 2:
                first, second, third = pos_pad[-3, 1]+1, pos_pad[-2, 1], pos_pad[-1, 1]
            else:
                # 有可能長度無須padding那起始即為0
                first, second, third = 0, pos_pad[-2, 1], pos_pad[-1, 1]

            # 找出哪個部份是prompt哪個是答案
            prompts.append(tokenizer.decode(inputs[n_batch, first: second]))
            answers.append(tokenizer.decode( inputs[n_batch, second: third] ))

        encoding = tokenizer(prompts, padding=True, return_tensors='pt').to('cuda')
        predicts = model.generate(**encoding, max_length=512, pad_token_id=tokenizer.pad_token_id)
        predicts = tokenizer.batch_decode(predicts, skip_special_tokens=True)

        for i, (pred, ans) in enumerate(zip(predicts, answers)):
            # 中文不是用空格當分割 -> 而是用一個一個字 -> list 切開 string
            candidate = list(pred[len(prompts[i])-len(tokenizer.eos_token):])
            ground_truth = [list(ans)]
            rouge.update(([candidate], [ground_truth]))

    return rouge.compute()

```

- Finetune 過程中我所擷取到的最佳gpt-2表現，可以發現比T5差非常多

Rouge-L-P	Rouge-L-R	Rouge-L-F	Rouge-2-P	Rouge-2-R	Rouge-2-F
0.187	0.179	0.179	0.116	0.109	0.109

- 範例生成結果：可以看出長度上不像T5知道要生成短文，短文內容也不太像摘要比較像接龍，甚至會有鬼打牆的行為瘋狂輸出某些字。

```
===== 8 =====
---> Prediction : *ST国创尚未收到此次立案
---> Ground Truth: 信披违规外加业绩亏损*ST国创退市风险概率大增
===== 9 =====
---> Prediction : 北京梅赛德斯-奔驰销售服务
---> Ground Truth: 发改委反垄断调查小组突击调查奔驰上海办事处
===== 10 =====
---> Prediction : 《查理周刊》袭击恐怖袭击
---> Ground Truth: 巴黎查理周刊最新一期封面(图)
===== 11 =====
---> Prediction : 深水区房地产调控政策走向研究包括土地
---> Ground Truth: 官媒吹风房地产调控长效机制：去行政化成主基调
===== 12 =====
---> Prediction : 上海女子学院：首批试点闵行区和长宁区学习中心
---> Ground Truth: 女子大学成立男性也可报名
===== 13 =====
---> Prediction : 白酒行业调整期内备受酒企热爱
---> Ground Truth: 龋战糖酒会小酒今年市场规模有望破100亿元
===== 14 =====
---> Prediction : 撰写学术论文和学术专著规则
---> Ground Truth: 掺水专著居然能获奖
===== 15 =====
---> Prediction : 沙特王子阿尔瓦利德·本·塔拉尔表示：沙特沙特沙特沙特沙特沙特沙特
---> Ground Truth: 沙特王子重申不减产美国页岩革命命悬一线
===== 16 =====
---> Prediction : 杭州明天小雨转阴：气温表气温渗透渗透渗透渗透渗透渗透渗透渗透渗透渗透渗透渗透渗透渗透渗透渗透渗透渗透
---> Ground Truth: 明后天有冷空气南下后天最高气温降到21°C
```

總結

- GPT-2訓練上較繁瑣，而且著重在下一個字的預測，因此表現差訓練時間長，尤其是在推論時更是耗時，整整比mT5慢了好幾倍
- mT5 這類的T5模型是一種encoder-decoder結構，因此，他可以非常容易切分出input和output，學習過程簡單且快速。同時，他可以透過encoder看過前文再利用decoder來生成任務所需，