

自然語言處理期末報告

# 網路用語的錯字修正 與語句改寫

資四B 10156209 李宓庭

資四B 10156230 陳奕蒨

資四B 10156233 陳苡璇

資四B 10156248 黃旗笙

資四B 10156259 陳竑華

第十二組

# 目錄

一、主題介紹

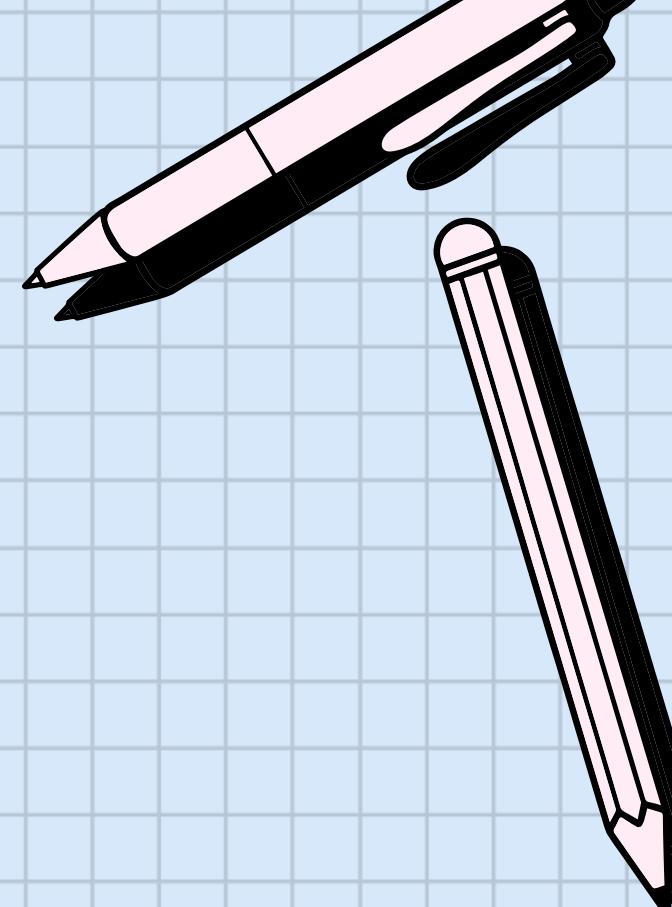
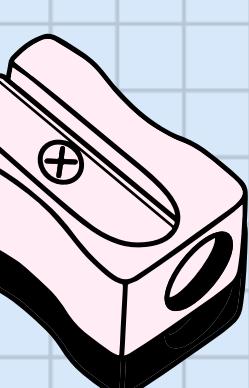
二、資料和方法

三、程式碼/成果展示

四、問題與討論

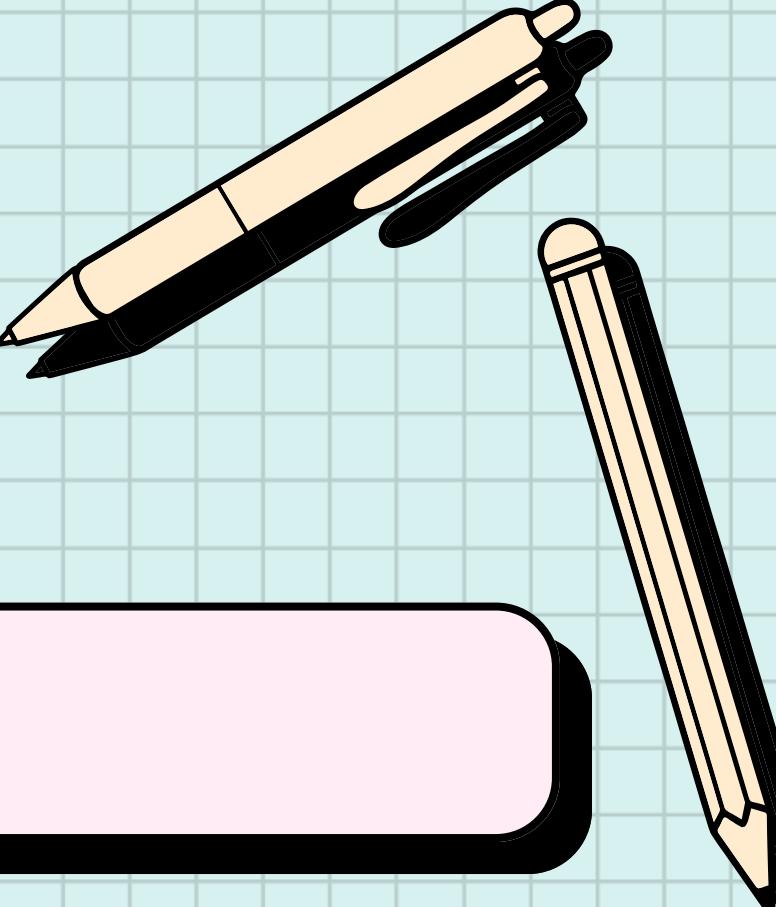
五、結論與感想

六、小組分工



# 主題介紹

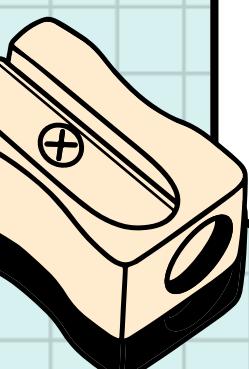
# 動機



## 網路用語的錯字修正與語句改寫

如今社群媒體平台上用詞縮寫、日常用語和流行用語隨處可見，這些網路用語變化快速、更新頻繁，常常令人一頭霧水。

而社群語言中也常出現病句、冗贅詞語等問題，因此我們希望建立一套能夠翻譯網路用語、修正常見病句與刪除冗詞贅字的系統。



# 目標

1. 準確翻譯與解釋網路用語的真實含義
2. 修正病句與口語化語句
3. 刪除冗贅詞語，使語句更清晰、自然
4. 文化差異修正（巨好吃 → 非常好吃）

# 資料和方法



# Process 1

## 蒐集資料集

- 收集1137筆input(原始輸入句)、target(修正句)
- 讓模型學習如何將網路用語、病句或冗贅詞的非標準句式，轉換為語意清楚、結構正確的標準句子

# Process 2

用BART, T5-small建立模型訓練資料集  
(因資料集為中文需改用bart-base-chinese)  
(google/mt5-small、uer/t5-base-chinese-cluecorp)

# Process 3

根據模型輸出結果，我們調整資料集內容與比例、優化樣本設計，並進行學習率、batch size 等參數微調，以提升模型表現。

# Why use BART/T5

|          | 句子生成            | 主要功能  | 優點  | 缺點  |
|----------|-----------------|---|---|---|
| BERT     | 無法<br>只能用在分類跟偵測 | <ul style="list-style-type: none"><li>病句位置偵測</li><li>文本分類</li></ul>               | <ul style="list-style-type: none"><li>適合偵錯任務</li><li>理解能力佳</li></ul>  | <ul style="list-style-type: none"><li>無法生成句子</li><li>無法改寫句子</li></ul> |
| BART     | 可以              | <ul style="list-style-type: none"><li>病句改寫</li><li>網路用語轉換</li><li>摘要/翻譯</li></ul> | <ul style="list-style-type: none"><li>文意理解佳</li><li>適合中大型任務</li></ul> | 訓練資源需求較高  |
| T5-small | 可以              | <ul style="list-style-type: none"><li>小資料集下的改寫任務</li><li>適合資源有限時使用</li></ul>      | 可在 CPU/GPU 上試驗  | 對複雜改寫較力不從心  |

# 程式碼/成果展示

# 程式碼

- 讀data.csv檔
- 切分訓練集和測試集為8:2
- 使用 bart-base-chinese 模型進行訓練

```
def main():
    # 讀取CSV
    data_path = os.path.join("data", "data0519.csv")
    df = pd.read_csv(data_path,
                      sep=",",
                      quotechar="'",
                      engine="python",
                      encoding="utf-8")
    if "_Unnamed: 0" in df.columns:
        df = df.drop(columns="_Unnamed: 0")
    print(df)
    # Dataset
    raw = Dataset.from_pandas(df, preserve_index=False)
    ds = raw.train_test_split(test_size=0.2) #訓練集測試集 8:2

    # 加載模型與分詞器
    model_name = "fnlp/bart-base-chinese"
    tokenizer = BertTokenizerFast.from_pretrained(model_name)
    model = BartForConditionalGeneration.from_pretrained(model_name)
```

# 程式碼

## 資料預處理

1. 對 input\_text 編碼
2. 對 target\_text 編碼
3. 將 labels 放進 inputs 中
4. Trainer 會自動抓取 labels

```
# 預處理
def preprocess(batch):
    inputs = tokenizer(batch["input_text"],
                      padding="max_length",
                      truncation=True,
                      max_length=128)
    inputs.pop("token_type_ids", None)

    labels = tokenizer(batch["target_text"],
                      padding="max_length",
                      truncation=True,
                      max_length=128)
    labels.pop("token_type_ids", None)

    inputs["labels"] = labels["input_ids"]
    return inputs

tokenized = ds.map(preprocess,
                   batched=True,
                   remove_columns=["input_text", "target_text"])
```

# 程式碼

## 設定訓練參數

```
training_args = TrainingArguments(  
    output_dir="bart_corrector", # 模型與檢查點儲存目錄  
    num_train_epochs=5, # 總訓練輪數 (完整遍歷訓練集的次數)  
    per_device_train_batch_size=5, # 每張 GPU 用於訓練的 batch 大小  
    per_device_eval_batch_size=4, # 每張 GPU 用於驗證的 batch 大小  
  
    do_train=True, # 啟用訓練階段  
    do_eval=True, # 啟用每步 / 每輪後的驗證  
    eval_steps=500, # 每 500 個步驟執行一次驗證  
    save_steps=500, # 每 500 個步驟儲存一次檢查點  
  
    logging_steps=50, # 每 50 個步驟輸出一次訓練日誌  
    learning_rate=2e-5, # 優化器的學習率  
    warmup_steps=100, # 前 100 步進行學習率 warm-up  
    weight_decay=0.01, # 權重衰減強度  
)
```

# 程式碼

- 整合到 Trainer
- 執行訓練

```
trainer = Trainer(  
    model=model,  
    args=training_args,  
    train_dataset=tokenized["train"],  
    eval_dataset=tokenized["test"],  
    tokenizer=tokenizer,  
    data_collator=custom_collator,  
    compute_metrics=compute_metrics  
)  
trainer.train()
```

# 程式碼

## 保存模型並預測結果

```
# 9. 保存模型
trainer.save_model("bart_corrector_best")
tokenizer.save_pretrained("bart_corrector_best")

# 10. 推理测试
def correct(text: str) -> str:
    device = model.device
    inputs = tokenizer(text,
                       return_tensors="pt",
                       padding="max_length",
                       truncation=True,
                       max_length=128).to(device)
    out_ids = model.generate(inputs.input_ids,
                             attention_mask=inputs.attention_mask,
                             max_length=128,
                             num_beams=4)
    return tokenizer.decode(out_ids[0], skip_special_tokens=True)

print("範例改寫：", correct("又來了啦煩欸"))
print("範例改寫：", correct("他是I人"))
print("範例改寫：", correct("我是什麼很賤的人嗎？為什麼來跟我借錢了？"))
print("範例改寫：", correct("他們給你跪了"))
print("範例改寫：", correct("干牠"))
print("範例改寫：", correct("我真的防房了"))

>>> if __name__ == "__main__":
        main()
```

# 輸出結果

```
print("範例改寫：", correct("乾你屁事"))
print("範例改寫：", correct("我快不行了 基督勞累"))
print("範例改寫：", correct("我不是有意的。"))
print("範例改寫：", correct("安安,哈囉"))
print("範例改寫：", correct("一起吃飯ㄉ我出錢"))
print("範例改寫：", correct("你真的很母湯"))
```

範例改寫： 關 你 什 麼 事  
範例改寫： 我 快 不 行 了 ， 只 能 勞 累 了 。  
範例改寫： 我 不 是 故 意 的 。  
範例改寫： 哈 囉 ， 你 好  
範例改寫： 一 起 吃 飯 吧 ， 我 出 錢 。  
範例改寫： 你 真 的 很 不 行

# 輸出結果

```
print("範例改寫：", correct("乾你屁事"))
print("範例改寫：", correct("你們端共"))
print("範例改寫：", correct("剛剛看到一個超好笑的影片笑吐了"))
print("範例改寫：", correct("他是yyds"))
print("範例改寫：", correct("你做的這件事我給87分，不能再高了"))
print("範例改寫：", correct("你真的很母湯"))
```

範例改寫： 關 你 什 麼 事

範例改寫： 你 們 出 來 說

範例改寫： 剛 剛 看 到 一 個 超 好 笑 的 影 片 ， 笑 到 不 行

範例改寫： 他 很 外 向

範例改寫： 你 做 這 件 事 真 的 很 白 癡

範例改寫： 你 真 的 很 不 行

# T5(mt5-small)

```
{'loss': 0.0, 'learning_rate': 3e-05, 'epoch': 19.88} ...
{'train_runtime': 295.684, 'train_samples_per_second': 48.768, 'train_steps_per_second': 0.338, 'train_loss': 0.0, 'epoch': 19.88}
100%[██████████] 100/100 [04:55<00:00, 2.70s/it]
```

原句：你這樣真的非常母湯

修正：`<pad> </s>`

原句：你們端共

修正：`<pad> </s>`

損失值始終為 0，顯示模型未能有效學習  
解決方法

- 調整 learning rate
- 關閉 gradient\_checkpointing
- 關閉 FP16 (半精度訓練)

# 問題與討論

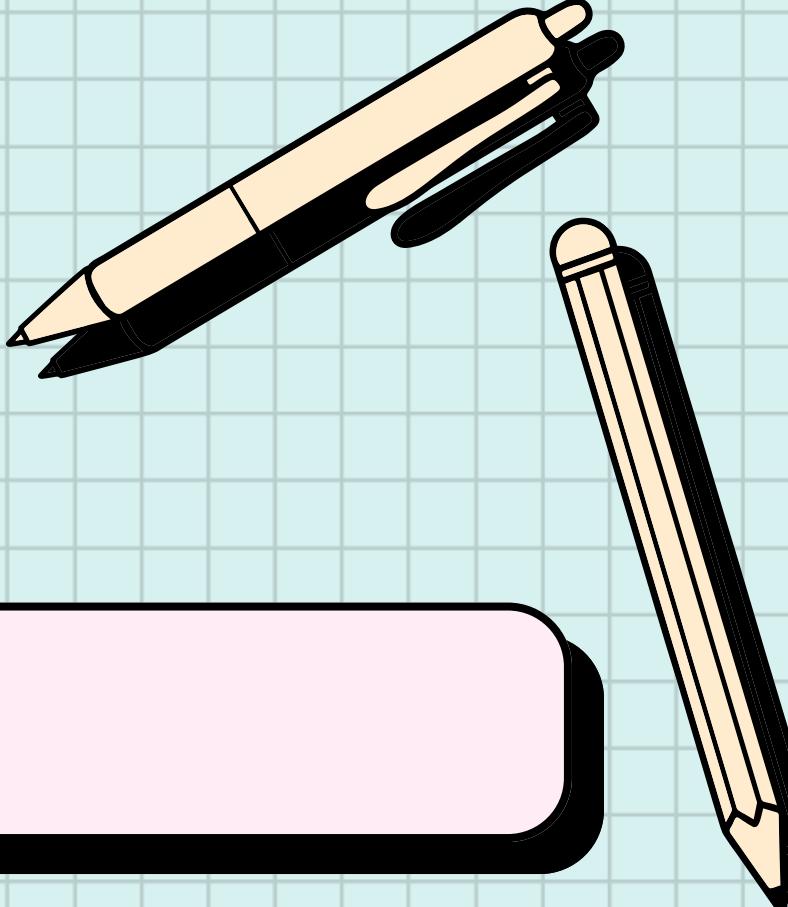
# 問題一

## 資料取得困難

- 發現網路上的資料不符合/沒有權限抓資料

原先打算使用 Facebook、Instagram 等網路平台上的開源資料進行實驗，但 API 權限限制嚴格、不易取得資料，且社交平台對自動爬蟲的限制高，資料存取效率低。因此後續轉向使用手動擴充的中文口語資料集與 ChatGPT 生成輔助。

## 問題二



### 模型語言衝突：英文/中文

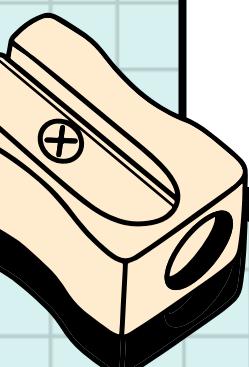
初期使用的 **facebook/bart-base** 為英文訓練模型，雖然能處理一般句子，但面對中文效果不佳。

改為 **fnlp/bart-base-chinese** 之後，整體訓練穩定性與結果品質大幅提升。這反映出模型語言訓練背景與資料語言需保持一致。

```
#初始化tokenizer
model_name = "facebook/bart-base"
tokenizer = BartTokenizer.from_pretrained(model_name)
model = BartForConditionalGeneration.from_pretrained(model_name)
```



```
model_name = "fnlp/bart-base-chinese"
tokenizer = BertTokenizerFast.from_pretrained(model_name)
model = BartForConditionalGeneration.from_pretrained(model_name)
```



# 問題三

## 樣本數太少

原本僅有數百筆樣本，無法讓模型有效學習錯字與語義轉換邏輯。

透過：

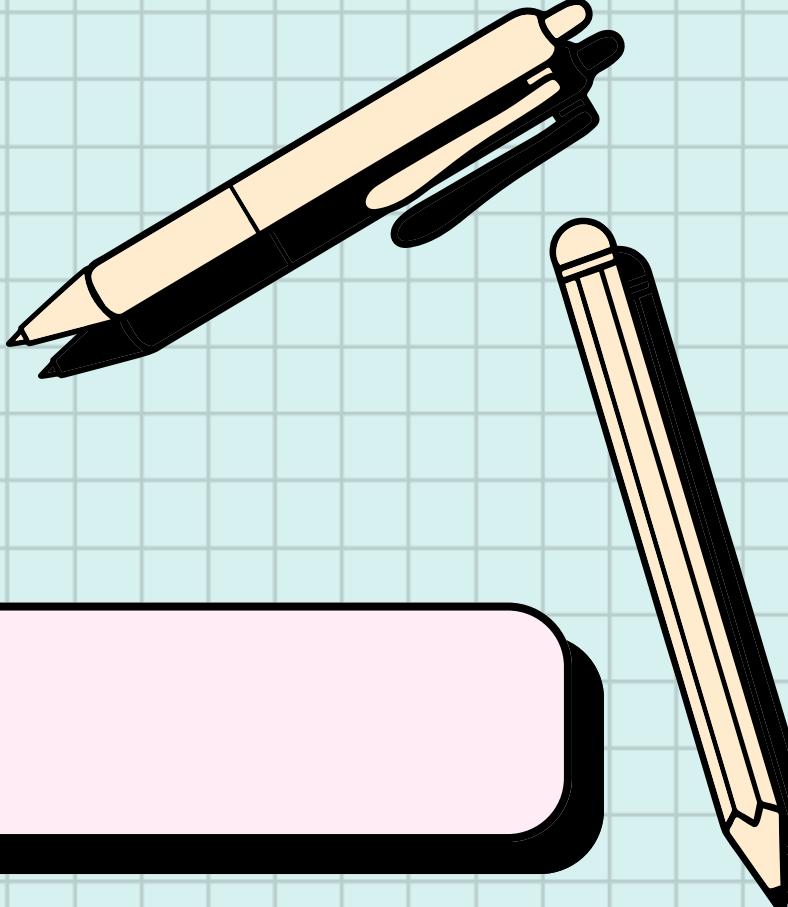
- 擴建流行語與錯誤句樣本
- 同義語句改寫：例如更換主詞、語氣或語序

Ex：「這家店真的很頂」→「他們那間店真的很頂」→「這間餐廳真的很厲害」

- 使用 GPT 輔助改寫正規句

擴增總數至 1137 筆才成功使 BART 模型順利收斂，提升訓練成果。

# 問題四



## Loss值跟訓練集/測試集分割比例間的關係①

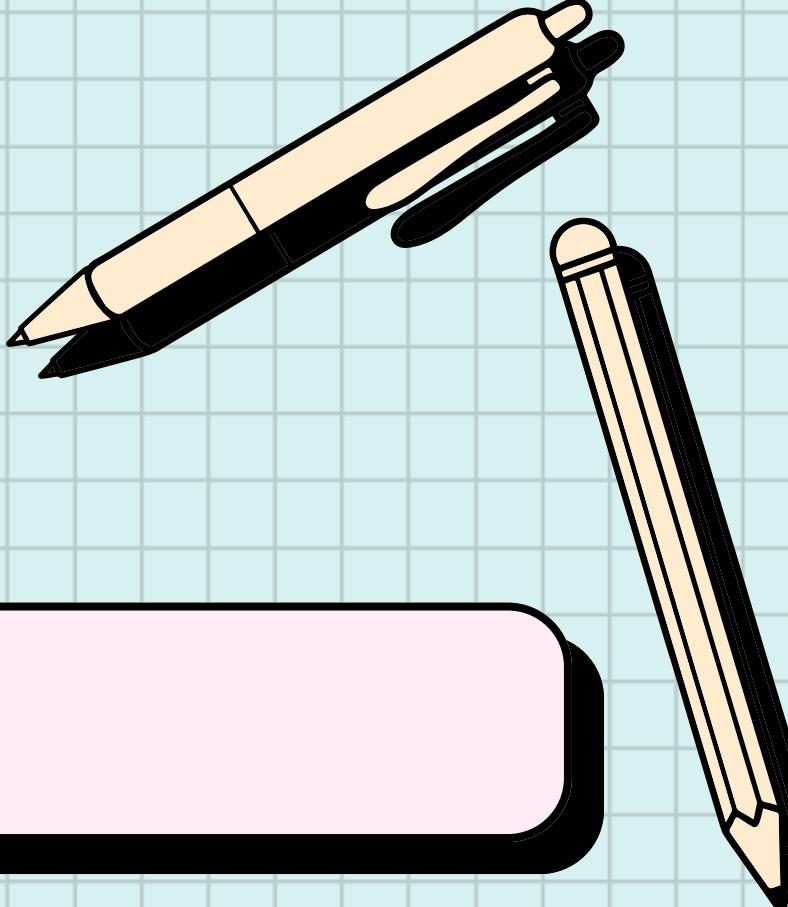
第一次切分：8:2（訓練集:測試集）

Loss 值：0.76

```
# Dataset  
raw = Dataset.from_pandas(df, preserve_index=False)  
ds  = raw.train_test_split(test_size=0.2) #訓練集測試集 8:2
```

```
'train_loss': 0.769566760227598,
```

# 問題四



## Loss值跟訓練集/測試集分割比例間的關係②

第二次切分：9:1（訓練集:測試集）

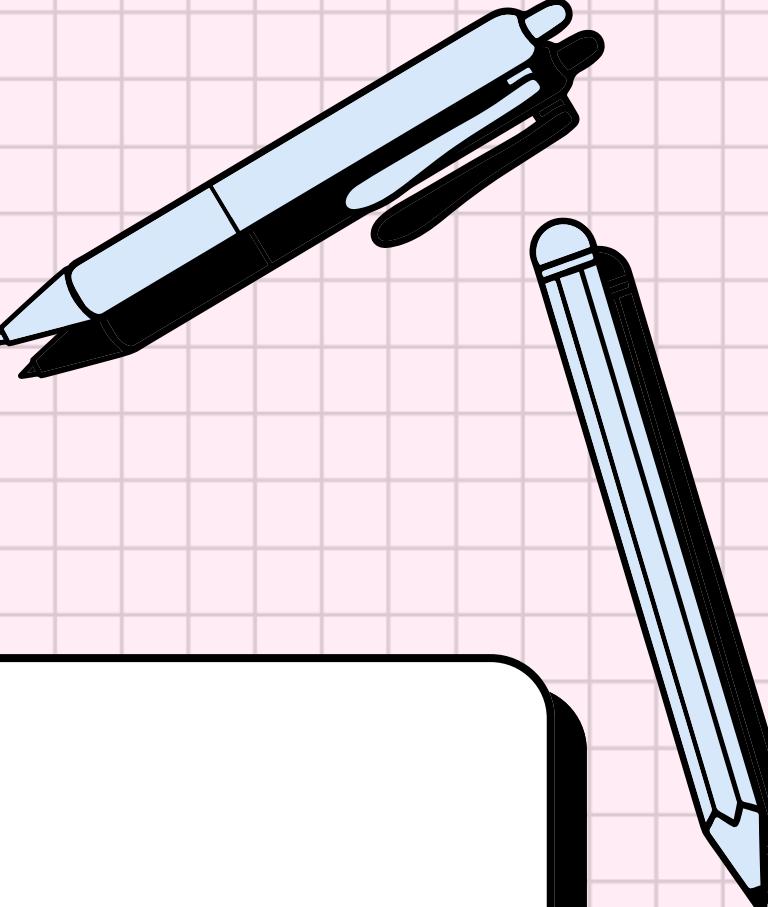
Loss 值：0.68

```
# Dataset  
raw = Dataset.from_pandas(df, preserve_index=False)  
ds  = raw.train_test_split(test_size=0.1) #訓練集測試集 8:1
```

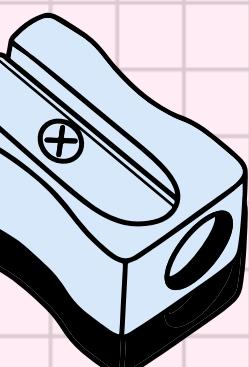
```
'train_loss': 0.6871160410664564,
```

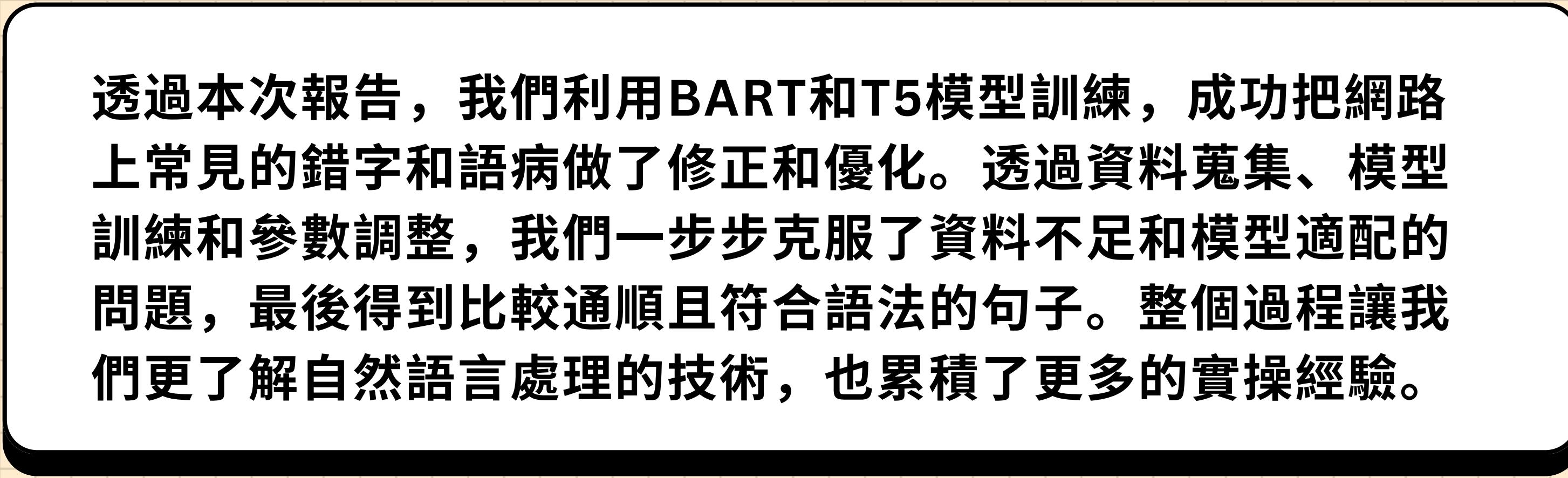
透過調整資料分配比例，有助於模型獲得更多訓練樣本，加速學習。





# 結論與感想





透過本次報告，我們利用BART和T5模型訓練，成功把網路  
上常見的錯字和語病做了修正和優化。透過資料蒐集、模型  
訓練和參數調整，我們一步步克服了資料不足和模型適配的  
問題，最後得到比較通順且符合語法的句子。整個過程讓我們  
更了解自然語言處理的技術，也累積了更多的實操經驗。

# 小組分工

**10156209**

**李宓庭**

- 主題發想
- 蒐集資料集
- 模型(BART)
- 模型(T5)

**10156230**

**陳奕蒨**

- 蒐集資料集
- PPT
- 書面報告

**10156233**

**陳苡璇**

- 主題發想
- 蒐集資料集
- PPT
- 模型(T5)

**10156248**

**黃旗笙**

- 蒐集資料集
- PPT
- 模型(T5)
- 上台報告

**10156259**

**陳竑華**

- 蒐集資料集
- PPT

**Thank  
you!**

