

DM2024-Lab2-Homework (/github/Iris6636/DM2024-Lab2-Homework/tree/main)

/

DM2024-Lab2-Homework.ipynb (/github/Iris6636/DM2024-Lab2-Homework/tree/main/DM2024-Lab2-Homework.ipynb)

## Student Information

Name: 吳君慧

Student ID: 113065539

GitHub ID: Iris6636

Kaggle name: Rondnoir

Kaggle private scoreboard snapshot:

41	← 2	Rondnoir		0.47377	4	4d
----	-----	----------	---	---------	---	----

## Instructions

1. First: **This part is worth 30% of your grade.** Do the **take home exercises** in the DM2024-Lab2-master Repo (<https://github.com/didiersalazar/DM2024-Lab2-Master>). You may need to copy some cells from the Lab notebook to this notebook.
2. Second: **This part is worth 30% of your grade.** Participate in the in-class Kaggle Competition (<https://www.kaggle.com/competitions/dm-2024-isa-5810-lab-2-homework>) regarding Emotion Recognition on Twitter by this link: <https://www.kaggle.com/competitions/dm-2024-isa-5810-lab-2-homework> (<https://www.kaggle.com/competitions/dm-2024-isa-5810-lab-2-homework>). The scoring will be given according to your place in the Private Leaderboard ranking:

- **Bottom 40%:** Get 20% of the 30% available for this section.
- **Top 41% - 100%:** Get  $(0.6N + 1 - x) / (0.6N) * 10 + 20$  points, where N is the total number of participants, and x is your rank. (ie. If there are 100 participants and you rank 3rd your score will be  $(0.6 * 100 + 1 - 3) / (0.6 * 100) * 10 + 20 = 29.67\%$  out of 30%.)

Submit your last submission **BEFORE the deadline (Nov. 26th, 11:59 pm, Tuesday)**. Make sure to take a screenshot of your position at the end of the competition and store it as "pic0.png" under the **img** folder of this repository and rerun the cell **Student Information**.

3. Third: **This part is worth 30% of your grade.** A report of your work developing the model for the competition (You can use code and comment on it). This report should include what your preprocessing steps, the feature engineering steps and an explanation of your model. You can also mention different things you tried and insights you gained.
4. Fourth: **This part is worth 10% of your grade.** It's hard for us to follow if your code is messy :(, so please **tidy up your notebook**.

Upload your files to your repository then submit the link to it on the corresponding e-learn assignment.

Make sure to commit and save your changes to your repository **BEFORE the deadline (Nov. 26th, 11:59 pm, Tuesday)**.

In [2]: `### Begin Assignment Here`

## SUMMARY

### 預處理

考量到推特文章的特性，例如包含 URL、@tag 用戶等，為避免這些元素對情緒分類的影響，設計了相應的前處理流程，目的是提供更乾淨且不易誤導的數據給 BERT 模型進行後續分析。

### 資料基本分析

#### 1. 情緒類別占比：

- 各情緒類別分佈差異較大，其中 `joy` 類別占比最高，明顯多於其他類別。

#### 2. 數據完整性：

- 經過檢查，`train set` 中沒有情緒標籤 (`emotion attribute`) 的空值情況。

### 模型選用

為了更好地捕捉推文中的情緒特徵，選用了基於 Transformer 架構的模型：

#### • 模型選擇原因：

- 使用 `distilbert-base-uncased` 模型，這是一個基於 BERT (`bert-base-uncased`) 的蒸餾版本，具備以下優勢：
  - 參數量僅為原模型的 60%，輕量化且高效。
  - 訓練速度與推論速度提升約 40%，非常適合硬體資源有限的環境。

## 實驗

### 實驗 1

1. 考量到各類別訓練資料不平均的狀況，採用 **undersampling**。

2. 藉由降低數據量，希望模型訓練時間縮短，能盡快得到一個成績作為 baseline。

3. 訓練參數：

- 訓練 3 個 epochs。

4. 實驗結果：

- Val set Macro F1 Score:** 0.4815
- Public leaderboard:** 0.42491
- Private leaderboard:** 0.41350

5. 評價：

- 得到了一個中偏下的排名與成績，可能是因為 `undersampling` 使數據量過小，模型無法有效學習。

## 實驗 2

1. 不進行 sampling，使用完整數據進行訓練。
2. 針對數據分佈不均的情況，將損失函數從 **CrossEntropyLoss** 改為 **Focal Loss**，以期更好應對類別不平衡的情況。
3. 訓練參數：
  - 訓練 3 個 epochs。
4. 實驗結果：
  - **Val set Macro F1 Score:** 0.5431（優於實驗 1）。
  - **Public leaderboard:** 0.48862
  - **Private leaderboard:** 0.47377
5. 評價：
  - 得到了一個中間排名與成績，數據量的增加讓模型學習到更多的特徵，提升了判斷能力。

## 可能的未來改進方向

- 模型改進：
  - 考慮使用參數量更大的模型，如 roberta-large 或其他更強的 Transformer 架構。
- 數據增強：
  - 嘗試對數據進行數據增強（Data Augmentation），提升模型對低頻類別的學習能力。
- Warm-up 策略：
  - 先用部分數據進行 Warm-up，幫助模型快速適應任務，建立更穩定的基礎。

```
In [1]: import pandas as pd
import numpy as np
import json
import re
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
import torch
```

## 1. 讀取資料

```
In [2]: import pandas as pd

# 讀取 JSON Lines 格式文件
tweets_df = pd.read_json('tweets_DM.json', lines=True)

# 查看前幾行
print(tweets_df.head())
```

	_score	_index	_source \
0	391	hashtag_tweets	{'tweet': {'hashtags': ['Snapchat'], 'tweet_id...
1	433	hashtag_tweets	{'tweet': {'hashtags': ['freepress', 'TrumpLeg...
2	232	hashtag_tweets	{'tweet': {'hashtags': ['bibleverse'], 'tweet...
3	376	hashtag_tweets	{'tweet': {'hashtags': [], 'tweet_id': '0x1cd5...
4	989	hashtag_tweets	{'tweet': {'hashtags': [], 'tweet_id': '0x2de2...

	_crawldate	_type
0	2015-05-23 11:42:47	tweets
1	2016-01-28 04:52:09	tweets
2	2017-12-25 04:39:20	tweets
3	2016-01-24 23:53:05	tweets
4	2016-01-08 17:18:59	tweets

```
In [3]: # 讀取 JSON，僅提取 tweet_id 和 text
tweets = []
with open('tweets_DM.json', 'r', encoding='utf-8') as file:
    for line in file:
        tweet_data = json.loads(line)
        tweet_id = tweet_data['_source']['tweet']['tweet_id']
        text = tweet_data['_source']['tweet']['text']
        tweets.append({'tweet_id': tweet_id, 'text': text})

tweets_df = pd.DataFrame(tweets)
print(tweets_df.head())
```

	tweet_id	text
0	0x376b20	People who post "add me on #Snapchat" must be ...
1	0x2d5350	@brianklaas As we see, Trump is dangerous to #...
2	0x28b412	Confident of your obedience, I write to you, k...
3	0x1cd5b0	Now ISSA is stalking Tasha 🤪🤪🤪 <LH>
4	0x2de201	"Trust is not the same as faith. A friend is s...

```
In [4]: # emotion.csv: 標籤資料
emotion = pd.read_csv('emotion.csv')

# data_identification.csv: 區分 train/test
data_identification = pd.read_csv('data_identification.csv')

df = data_identification.merge(tweets_df, on='tweet_id', how='left')
df = df.merge(emotion, on='tweet_id', how='left')
print(df.head())
```

	tweet_id	identification	text \
0	0x28cc61	test	@Habbo I've seen two separate colours of the e...
1	0x29e452	train	Huge Respect👏 @JohnnyVegasReal talking about l...
2	0x2b3819	train	Yoooo we hit all our monthly goals with the ne...
3	0x2db41f	test	@FoxNews @KellyannePolls No serious self respe...
4	0x2a2acc	train	@KIDSNTS @PICU_BCH @uhbcomms @BWCHBoss Well do...

	emotion
0	NaN
1	joy
2	joy
3	NaN
4	trust

```
In [5]: # 確認 `emotion` 欄位中各類別的數量
emotion_counts = df['emotion'].value_counts()

print("各情感類別的數量:")
print(emotion_counts)
```

```
各情感類別的數量:
emotion
joy          516017
anticipation 248935
trust        205478
sadness      193437
disgust      139101
fear         63999
surprise     48729
anger        39867
Name: count, dtype: int64
```

--> 可以看到，各情緒類別的占比其實差異滿大的，joy 占了很大部分

--> 有確認 train set 中沒有情緒 attribute 是空值的狀況

```
In [9]: type_counts = df['identification'].value_counts()
```

```
print("各情感類別的數量：")
print(type_counts)
```

```
各情感類別的數量：
identification
train    1455563
test      411972
Name: count, dtype: int64
```

--> 可以看到，train 的數據量很大，應該可以給模型很多樣的資訊

```
In [6]: df.head(20)
```

```
Out[6]:
```

	tweet_id	identification	text	emotion
0	0x28cc61	test	@Habbo I've seen two separate colours of the e...	NaN
1	0x29e452	train	Huge Respect👏 @JohnnyVegasReal talking about l...	joy
2	0x2b3819	train	Yoooo we hit all our monthly goals with the ne...	joy
3	0x2db41f	test	@FoxNews @KellyannePolls No serious self respe...	NaN
4	0x2a2acc	train	@KIDSNTS @PICU_BCH @uhbcomms @BWCHBoss Well do...	trust
5	0x2a8830	train	Come join @ambushman27 on #PUBG while he striv...	joy
6	0x20b21d	train	@fanshixieen2014 Blessings!My #strength little...	anticipation
7	0x2452cf	train	Never give up. The manifestation of your goal ...	anticipation
8	0x2d729d	train	I Believe When No One Else Does... <LH> #Dream...	anticipation
9	0x2ab56d	train	@SirPareshRawal with due respect... Do u have ...	joy
10	0x1f3657	train	I can't tell if I'm alive or in the after life...	sadness
11	0x1fcc53	train	#GRATEFUL!! WORLD GOODMORNING!	trust
12	0x254da9	train	@happinessplannr #LovelsLove <LH> this!! 💕👫	joy
13	0x2e19fe	train	Watching Santa Claus with my parents❤️ <LH>	joy
14	0x37c6da	train	Encounters with the <LH> of God will change yo...	joy
15	0x2466f6	test	Looking for a new car, and it says 1 lady owne...	NaN
16	0x2c3d04	train	95° or higher 17 out of the last 19 days, no r...	disgust
17	0x1e414d	train	#Congratulations to all our #Graduates from @S...	trust
18	0x1f0ab5	train	Hustle Season Has Return <LH>	anticipation
19	0x275fcc	train	3nights out this weekend has ruined me. Twiste...	sadness

## 2. 資料清理

定義清理函數

考量到推特文章會有一些特性，像是包含URL、tag 其他使用者等等，故作了一個相對應的前處理功能，以提供一個比較不會有太誤導狀況的資料給BERT做後續分析

```
In [10]: import re
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer

# 下載必要的 NLTK 資源
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')

# 初始化停用詞集合和詞形還原器
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()

def preprocess_for_bert(tweet):
    # 1. 移除 URL
    tweet = re.sub(r'http\S+|www\S+|https\S+', '', tweet, flags=re.MULTILINE)
    # 2. 移除用戶提及
    tweet = re.sub(r'@\w+', '', tweet)
    # 3. 移除 Emoji
    tweet = tweet.encode('ascii', 'ignore').decode('ascii')
    # 4. 保留標籤內容 · 移除 `#`
    tweet = re.sub(r'#', '', tweet)
    # 5. 統一轉換為小寫 (適用於 uncased 模型)
    tweet = tweet.lower()
    return tweet

# 範例推文
tweet = "@user I absolutely LOVE the new #iPhone12!!! It's soooo amazing! Check it out at ht
processed_tweet = preprocess_for_bert(tweet)
print("原始推文:", tweet)
print("BERT 前處理後的推文:", processed_tweet)
```

原始推文: @user I absolutely LOVE the new #iPhone12!!! It's soooo amazing! Check it out at https://apple.com  
 BERT 前處理後的推文: i absolutely love the new iphone12!!! it's soooo amazing! check it out at

```
[nltk_data] Downloading package punkt to
[nltk_data] C:\Users\thpss\AppData\Roaming\nltk_data...
[nltk_data] Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data] C:\Users\thpss\AppData\Roaming\nltk_data...
[nltk_data] Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data] C:\Users\thpss\AppData\Roaming\nltk_data...
[nltk_data] Package wordnet is already up-to-date!
```

```
In [11]: df['cleaned_text'] = df['text'].apply(preprocess_for_bert)
print(df[['text', 'cleaned_text']].head())
```

```
text \
0 @Habbo I've seen two separate colours of the e...
1 Huge Respect👏 @JohnnyVegasReal talking about l...
2 Yoooo we hit all our monthly goals with the ne...
3 @FoxNews @KellyannePolls No serious self respe...
4 @KIDSNTS @PICU_BCH @uhbcomms @BWCHBoss Well do...

cleaned_text
0 i've seen two separate colours of the elegant...
1 huge respect talking about losing his dad to ...
2 yoooo we hit all our monthly goals with the ne...
3 no serious self respecting individual believ...
4 well done team <lh> of every one of you.
```

```
In [12]: df.head()
```

Out[12]:	tweet_id	identification	text	emotion	cleaned_text
0	0x28cc61	test	@Habbo I've seen two separate colours of the e...	NaN	i've seen two separate colours of the elegant...
1	0x29e452	train	Huge Respect👏 @JohnnyVegasReal talking about l...	joy	huge respect talking about losing his dad to ...
2	0x2b3819	train	Yoooo we hit all our monthly goals with the ne...	joy	yoooo we hit all our monthly goals with the ne...
3	0x2db41f	test	@FoxNews @KellyannePolls No serious self respe...	NaN	no serious self respecting individual believ...
4	0x2a2acc	train	@KIDSNTS @PICU_BCH @uhbcomms @BWCHBoss Well do...	trust	well done team <lh> of every one of you.

前處理完成，將 train & test set 分開

```
In [13]: # 分割成 train 和 test 的 DataFrame
train_df_ttl = df[df['identification'] == 'train']
test_df = df[df['identification'] == 'test']

# 檢查分割後的結果
print(f"Train DataFrame 大小: {train_df_ttl.shape}")
print(f"Test DataFrame 大小: {test_df.shape}")
```

Train DataFrame 大小: (1455563, 5)

Test DataFrame 大小: (411972, 5)

### 3. 使用 BERT 抓取特徵並進行分類

```
In [14]: print("Using GPU" if torch.cuda.is_available() else "Using CPU")
```

Using GPU

#### 模型選用: distilbert-base-uncased

選用原因: 因為手邊的硬體資源較不足，經查詢"distilbert-base-uncased" 是一基於 BERT (bert-base-uncased) 的蒸餾版本，其參數量較小，且訓練跟推論數度都更快一些。

#### 實驗1:

考量到各類別訓練資料不平均的狀況，採用 undersampling

同時也藉由降低數據量，希望模型不要 train 太久，能盡快看到一個成績當作 baseline

```

In [ ]: from sklearn.model_selection import train_test_split
from transformers import DistilBertTokenizer, DistilBertForSequenceClassification
from sklearn.preprocessing import LabelEncoder
from torch.utils.data import DataLoader, Dataset
import torch
import torch.nn as nn
import torch.optim as optim
from tqdm import tqdm
from sklearn.metrics import f1_score
from sklearn.utils import resample
import numpy as np

# 1. 確認資料格式
print("準備資料中...")
print(train_df_ttl.head())

# 2. Label Encoding
label_encoder = LabelEncoder()
train_df_ttl['label'] = label_encoder.fit_transform(train_df_ttl['emotion']) # 把情感文字轉換成數字

# 3. Train-Test Split
train_df, val_df = train_test_split(train_df_ttl, test_size=0.2, stratify=train_df_ttl['label'])
print(f"訓練集大小: {len(train_df)}, 驗證集大小: {len(val_df)}")

# 4. Under Sampling 函數
def undersample_data(df, label_col):
    min_count = df[label_col].value_counts().min() # 最小樣本數
    balanced_dfs = []
    for label in df[label_col].unique():
        class_subset = df[df[label_col] == label]
        balanced_dfs.append(resample(class_subset, replace=False, n_samples=min_count, random_state=42))
    balanced_df = pd.concat(balanced_dfs)
    return balanced_df

# 進行 Under Sampling
train_df_balanced = undersample_data(train_df, 'label')
print(f"平衡後的訓練集大小: {len(train_df_balanced)}")

# 5. 使用 DistilBERT 的 tokenizer
tokenizer = DistilBertTokenizer.from_pretrained('distilbert-base-uncased')

# 6. 定義數據集處理類別
class EmotionDataset(Dataset):
    def __init__(self, dataframe, tokenizer, max_len=128):
        self.dataframe = dataframe
        self.tokenizer = tokenizer
        self.max_len = max_len

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

    def __getitem__(self, idx):
        row = self.dataframe.iloc[idx]
        text = row['cleaned_text']
        label = row['label']
        encoding = self.tokenizer(
            text,
            truncation=True,
            padding='max_length',
            max_length=self.max_len,
            return_tensors='pt'
        )
        item = {key: val.squeeze(0) for key, val in encoding.items()}
        item['labels'] = torch.tensor(label, dtype=torch.long)

```



```
        return item

# 建立數據集與 DataLoader
train_dataset = EmotionDataset(train_df_balanced, tokenizer)
val_dataset = EmotionDataset(val_df, tokenizer)
train_loader = DataLoader(train_dataset, batch_size=16, shuffle=True)
val_loader = DataLoader(val_dataset, batch_size=64)

# 7. 初始化模型
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = DistilBertForSequenceClassification.from_pretrained('distilbert-base-uncased', num_labels=5)
model = model.to(device)

# 8. 定義損失函數和優化器
criterion = nn.CrossEntropyLoss()
optimizer = optim.AdamW(model.parameters(), lr=5e-5)

# 9. 訓練函數
def train_epoch(model, dataloader, optimizer, criterion, device):
    model.train()
    epoch_loss = 0
    progress_bar = tqdm(dataloader, desc="Training", leave=False)
    for batch in progress_bar:
        inputs = {key: val.to(device) for key, val in batch.items() if key != 'labels'}
        labels = batch['labels'].to(device)

        optimizer.zero_grad()
        outputs = model(**inputs)
        loss = criterion(outputs.logits, labels)
        loss.backward()
        optimizer.step()

        epoch_loss += loss.item()
        progress_bar.set_postfix(loss=loss.item())
    return epoch_loss / len(dataloader)

# 10. 驗證函數 (計算 Macro F1 Score)
def eval_model(model, dataloader, criterion, device):
    model.eval()
    epoch_loss = 0
    all_preds = []
    all_labels = []
    progress_bar = tqdm(dataloader, desc="Evaluating", leave=False)
    with torch.no_grad():
        for batch in progress_bar:
            inputs = {key: val.to(device) for key, val in batch.items() if key != 'labels'}
            labels = batch['labels'].to(device)

            outputs = model(**inputs)
            loss = criterion(outputs.logits, labels)
            epoch_loss += loss.item()

            preds = torch.argmax(outputs.logits, dim=1)
            all_preds.extend(preds.cpu().numpy())
            all_labels.extend(labels.cpu().numpy())

        progress_bar.set_postfix(loss=loss.item())
    f1 = f1_score(all_labels, all_preds, average='macro')
    return epoch_loss / len(dataloader), f1

# 11. 主訓練迴圈
epochs = 3
for epoch in range(epochs):
    print(f"Epoch {epoch + 1}/{epochs}")
    # 執行訓練
    train_loss = train_epoch(model, train_loader, optimizer, criterion, device)
```

```
# 執行驗證
val_loss, val_f1 = eval_model(model, val_loader, criterion, device)

# 輸出訓練與驗證結果
print(f"Training Loss: {train_loss:.4f}, Validation Loss: {val_loss:.4f}, Validation Macro F1 Score: {val_f1:.4f}")

# 保存模型
epoch_dir = f'./distilbert_emotion_model_{epoch + 1}'
model.save_pretrained(epoch_dir)
tokenizer.save_pretrained(epoch_dir)
print(f"模型已保存至 {epoch_dir}!")
```

```
訓練集大小: 1164450, 驗證集大小: 291113
平衡後的訓練集大小: 255152

Some weights of DistilBertForSequenceClassification were not initialized from the model checkpoint at ./distilbert-base-uncased and are newly initialized from the normal distribution.
['classifier.bias', 'classifier.weight', 'pre_classifier.bias', 'pre_classifier.weight']
You should probably TRAIN this model on a down-stream task to be able to use it for inference.

Epoch 1/3

Training Loss: 1.3923, Validation Loss: 1.2687, Validation Macro F1 Score: 0.4620
模型已保存至 ./distilbert_emotion_model_1!
Epoch 2/3

Training Loss: 1.1869, Validation Loss: 1.3367, Validation Macro F1 Score: 0.4619
模型已保存至 ./distilbert_emotion_model_2!
Epoch 3/3

Training Loss: 1.0080, Validation Loss: 1.3057, Validation Macro F1 Score: 0.4815
模型已保存至 ./distilbert_emotion_model_3!
```

總共訓練了3個 epochs,因為第3個 epoch 的表現最好,故繼續往下進行test set 預測,沒有額外再切換模型。

## 4. 進行 Test Set 預測

```
In [ ]: # 1. 定義測試數據集處理類別
class TestDataset(Dataset):
    def __init__(self, dataframe, tokenizer, max_len=128):
        self.dataframe = dataframe
        self.tokenizer = tokenizer
        self.max_len = max_len

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

    def __getitem__(self, idx):
        row = self.dataframe.iloc[idx]
        text = row['cleaned_text']
        encoding = self.tokenizer(
            text,
            truncation=True,
            padding='max_length',
            max_length=self.max_len,
            return_tensors='pt'
        )
        item = {key: val.squeeze(0) for key, val in encoding.items()}
        return item

# 2. 構建測試 DataLoader
test_dataset = TestDataset(test_df, tokenizer)
test_loader = DataLoader(test_dataset, batch_size=64)

# 3. 預測函數
def predict_model(model, dataloader, device):
    model.eval()
    predictions = []
    progress_bar = tqdm(dataloader, desc="Predicting", leave=False)
    with torch.no_grad():
        for batch in progress_bar:
            inputs = {key: val.to(device) for key, val in batch.items()}
            outputs = model(**inputs)
            preds = torch.argmax(outputs.logits, dim=1)
            predictions.extend(preds.cpu().numpy())
    return predictions

# 4. 對測試數據進行預測
print("開始對測試數據進行預測...")
test_predictions = predict_model(model, test_loader, device)

# 5. 將預測結果轉換為原始情感標籤
test_df['predicted_emotion'] = label_encoder.inverse_transform(test_predictions)

# 6. 儲存預測結果
'''
submission = test_df[['tweet_id', 'predicted_emotion']] # 只保留 tweet_id 和 emotion 欄位
submission.to_csv('submission.csv', index=False) # 儲存為 submission.csv
print("測試數據的預測結果已保存為 'submission.csv'")
'''

# 重命名欄位
submission = test_df[['tweet_id', 'predicted_emotion']].rename(columns={
    'tweet_id': 'id',
    'predicted_emotion': 'emotion'
})

# 儲存為 submission.csv
submission.to_csv('submission.csv', index=False)
print("測試數據的預測結果已保存為 'submission.csv'")
```

此預測結果獲得 public score: 0.42491

算是一個中偏下的排名與成績

可能是因為undersampling 讓數據量變得太小，可能模型不太好學習

	submission-f1.csv Complete · 4d ago · undersample+f1 training	private	public
		0.41350	0.42491

## 實驗2:

不做sampling, 使用全部的數據去訓練

但依舊考量到數據量不平均的狀況，故將損失評估從 CrossEntropyLoss 改成 focal loss，以期更好應對類別不平衡情況。

```
In [ ]: import pandas as pd
from sklearn.model_selection import train_test_split
from transformers import DistilBertTokenizer, DistilBertForSequenceClassification
from sklearn.preprocessing import LabelEncoder
from torch.utils.data import DataLoader, Dataset
import torch
import torch.nn as nn
import torch.optim as optim
from tqdm import tqdm
from torch.nn import functional as F
from sklearn.metrics import f1_score

# 1. 確認資料格式
print("準備資料中...")
print(train_df_ttl.head())

# Label Encoding 情感標籤
label_encoder = LabelEncoder()
train_df_ttl['label'] = label_encoder.fit_transform(train_df_ttl['emotion']) # 把情感文字轉換為數字

# 3. Train-Test Split
train_df, val_df = train_test_split(train_df_ttl, test_size=0.2, stratify=train_df_ttl['label'])
print(f"訓練集大小: {len(train_df)}, 驗證集大小: {len(val_df)}")

# 4. 使用 DistilBERT 的 tokenizer
tokenizer = DistilBertTokenizer.from_pretrained('distilbert-base-uncased')

# 5. 定義數據集處理類別
class EmotionDataset(Dataset):
    def __init__(self, dataframe, tokenizer, max_len=128):
        self.dataframe = dataframe
        self.tokenizer = tokenizer
        self.max_len = max_len

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

    def __getitem__(self, idx):
        row = self.dataframe.iloc[idx]
        text = row['cleaned_text']
        label = row['label']
        encoding = self.tokenizer(
            text,
            truncation=True,
            padding='max_length',
            max_length=self.max_len,
            return_tensors='pt'
        )
        item = {key: val.squeeze(0) for key, val in encoding.items()}
        item['labels'] = torch.tensor(label, dtype=torch.long)
        return item

train_dataset = EmotionDataset(train_df, tokenizer)
val_dataset = EmotionDataset(val_df, tokenizer)

# 6. 構建 DataLoader
train_loader = DataLoader(train_dataset, batch_size=16, shuffle=True)
val_loader = DataLoader(val_dataset, batch_size=64)

# 7. 初始化模型
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = DistilBertForSequenceClassification.from_pretrained('distilbert-base-uncased', num_labels=4)
model = model.to(device)

# 8. 定義 Focal Loss
```

```
class FocalLoss(nn.Module):
    def __init__(self, alpha=1, gamma=2):
        super(FocalLoss, self).__init__()
        self.alpha = alpha
        self.gamma = gamma

    def forward(self, logits, targets):
        ce_loss = F.cross_entropy(logits, targets, reduction="none")
        pt = torch.exp(-ce_loss) # Probabilities for true labels
        focal_loss = self.alpha * (1 - pt) ** self.gamma * ce_loss
        return focal_loss.mean()

criterion = FocalLoss() # 使用 Focal Loss
optimizer = optim.AdamW(model.parameters(), lr=5e-5)

# 9. 訓練函數
def train_epoch(model, dataloader, optimizer, criterion, device):
    model.train()
    epoch_loss = 0
    progress_bar = tqdm(dataloader, desc="Training", leave=False)
    for batch in progress_bar:
        inputs = {key: val.to(device) for key, val in batch.items() if key != 'labels'}
        labels = batch['labels'].to(device)

        optimizer.zero_grad()
        outputs = model(**inputs)
        loss = criterion(outputs.logits, labels)
        loss.backward()
        optimizer.step()

        epoch_loss += loss.item()
        progress_bar.set_postfix(loss=loss.item())
    return epoch_loss / len(dataloader)

# 10. 驗證函數 (使用 F1 Score)
def eval_model(model, dataloader, device):
    model.eval()
    predictions = []
    true_labels = []
    progress_bar = tqdm(dataloader, desc="Evaluating", leave=False)
    with torch.no_grad():
        for batch in progress_bar:
            inputs = {key: val.to(device) for key, val in batch.items() if key != 'labels'}
            labels = batch['labels'].to(device)

            outputs = model(**inputs)
            preds = torch.argmax(outputs.logits, dim=1)
            predictions.extend(preds.cpu().numpy())
            true_labels.extend(labels.cpu().numpy())

    f1 = f1_score(true_labels, predictions, average='macro')
    return f1

# 11. 主訓練迴圈
epochs = 3
for epoch in range(epochs):
    print(f"Epoch {epoch + 1}/{epochs}")
    # 執行訓練
    train_loss = train_epoch(model, train_loader, optimizer, criterion, device)
    # 執行驗證
    val_f1 = eval_model(model, val_loader, device)

    # 輸出訓練與驗證結果
    print(f"Training Loss: {train_loss:.4f}, Validation F1: {val_f1:.4f}")

    # 保存模型
```

```
epoch_dir = f'./distilbert_emotion_model_{epoch + 1}'  
model.save_pretrained(epoch_dir)  
tokenizer.save_pretrained(epoch_dir)  
print(f"模型已保存至 {epoch_dir}!")
```

訓練集大小: 1164450, 驗證集大小: 291113

Some weights of DistilBertForSequenceClassification were not initialized from the pretrained state dictionary. You should probably TRAIN this model on a down-stream task to initialize them with useful parameters.

Epoch 1/3

Training Loss: 0.7150, Validation F1: 0.5288

模型已保存至 ./distilbert\_emotion\_model\_1!

Epoch 2/3

Training Loss: 0.6222, Validation F1: 0.5328

模型已保存至 ./distilbert\_emotion\_model\_2!

Epoch 3/3

Training Loss: 0.5683, Validation F1: 0.5431

模型已保存至 ./distilbert\_emotion\_model\_3!

在訓練中的 F1 驗證成績已經看到比實驗1還要好

一樣，總共訓練了3個 epochs,因為第3個 epoch 的表現最好，故繼續往下進行test set 預測，沒有額外再切換模型。

做Test Set 預測



```

In [ ]: # 1. 定義測試數據集處理類別
class TestDataset(Dataset):
    def __init__(self, dataframe, tokenizer, max_len=128):
        self.dataframe = dataframe
        self.tokenizer = tokenizer
        self.max_len = max_len

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

    def __getitem__(self, idx):
        row = self.dataframe.iloc[idx]
        text = row['cleaned_text']
        encoding = self.tokenizer(
            text,
            truncation=True,
            padding='max_length',
            max_length=self.max_len,
            return_tensors='pt'
        )
        item = {key: val.squeeze(0) for key, val in encoding.items()}
        return item

# 2. 構建測試 DataLoader
test_dataset = TestDataset(test_df, tokenizer)
test_loader = DataLoader(test_dataset, batch_size=64)

# 3. 預測函數
def predict_model(model, dataloader, device):
    model.eval()
    predictions = []
    progress_bar = tqdm(dataloader, desc="Predicting", leave=False)
    with torch.no_grad():
        for batch in progress_bar:
            inputs = {key: val.to(device) for key, val in batch.items()}
            outputs = model(**inputs)
            preds = torch.argmax(outputs.logits, dim=1)
            predictions.extend(preds.cpu().numpy())
    return predictions

# 4. 對測試數據進行預測
print("開始對測試數據進行預測...")
test_predictions = predict_model(model, test_loader, device)

# 5. 將預測結果轉換為原始情感標籤
test_df['predicted_emotion'] = label_encoder.inverse_transform(test_predictions)

# 6. 儲存預測結果
'''
submission = test_df[['tweet_id', 'predicted_emotion']] # 只保留 tweet_id 和 emotion 欄位
submission.to_csv('submission.csv', index=False) # 儲存為 submission.csv
print("測試數據的預測結果已保存為 'submission.csv'")
'''


# 重命名欄位
submission = test_df[['tweet_id', 'predicted_emotion']].rename(columns={
    'tweet_id': 'id',
    'predicted_emotion': 'emotion'
})

# 儲存為 submission.csv
submission.to_csv('submission.csv', index=False)
print("測試數據的預測結果已保存為 'submission.csv'")

```

此預測結果獲得 public score: 0.48862

算是一個中間的排名與成績

Submission and Description		Private Score ⓘ	Public Score ⓘ
	<b>submission-destilbert_f1_3epoch.csv</b> Complete · 4d ago	<b>0.47377</b>	<b>0.48862</b>