Student Information

Name: 張子凡

Student ID: 112139503

GitHub ID: spikey09

Kaggle name: spikeyscii

Kaggle private scoreboard snapshot:

picture

Instructions

1. First: **This part is worth 30% of your grade.** Do the **take home exercises** in the DM2024-Lab2-master Repo. 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 regarding Emotion Recognition on Twitter by this link: 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)**.

Install package

Automatically check and install package for quckily deploy on any local device.

```
import subprocess
import sys

# 函式清單，包含需要檢查的函式庫名稱（可以指定版本）
libraries = {
    "pandas": None,  # 最新版本
    "numpy": None,
    "nltk": None,
    "matplotlib": None,
    "seaborn": None,
    "itertools": None,  # itertools 是內建模組，無需安裝
    "umap-learn": None,
    "gensim": None,
    "tensorflow": None,
    "keras": None,
    "ollama": None,
    "langchain": None,
    "langchain_community": None,
    "langchain_core": None,
    "bs4": None,
    "chromadb": None,
    "gradio": None,
    "emoji": None  # 指定版本
}

# 檢查函式庫是否已安裝，若未安裝則自動安裝
for lib, version in libraries.items():
    try:
        if lib == "itertools":
            # itertools 是內建模組，直接跳過
            continue

        # 若有指定版本，檢查該版本是否已安裝
        if version:
            import pkg_resources
            pkg_resources.require(f"{lib}=={version}")
            print(f"{lib}=={version} is already installed.")
        else:
            # 若無版本限制，只檢查模組是否存在
            __import__(lib)
            print(f"{lib} is already installed.")
    except ImportError:
```

```
        # 安装指定版本或最新版本
        print(f"{lib} is not installed. Installing...")
        try:
            if version:
                subprocess.check_call([sys.executable, "-m", "pip", 
"install", f"{lib}=={version}"])
            else:
                subprocess.check_call([sys.executable, "-m", "pip", 
"install", lib])
        except Exception as e:
            print(f"Failed to install {lib}: {e}")
    except pkg_resources.VersionConflict as e:
        # 如果版本不符，重新安装指定版本
        print(f"Version conflict for {lib}. Reinstalling version 
{version}...")
        try:
            subprocess.check_call([sys.executable, "-m", "pip", 
"install", f"{lib}=={version}"])
        except Exception as e:
            print(f"Failed to install {lib}: {e}")

print("All libraries checked.")
```

```
pandas is already installed.
numpy is already installed.
nltk is already installed.
matplotlib is already installed.
seaborn is already installed.
umap-learn is not installed. Installing...
gensim is already installed.
tensorflow is already installed.
keras is already installed.
ollama is already installed.
langchain is already installed.
langchain_community is already installed.
langchain_core is already installed.
bs4 is already installed.
chromadb is already installed.
gradio is already installed.
emoji is already installed.
All libraries checked.
```

```
!nvidia-smi
```

```
Mon Nov 25 20:04:02 2024       
+-----------------------------------------------------------------------
--------------------+
| NVIDIA-SMI 560.94                 Driver Version: 560.94         
CUDA Version: 12.6     |
|-----------------------------------------+------------------------
```

```
+---------------------+
| GPU  Name                  Driver-Model | Bus-Id          Disp.A |
Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage |
GPU-Util  Compute M. |
|                                         |                        |
MIG M. |
|
=========================================+========================+===
===================|
|   0  NVIDIA GeForce RTX 4080 ...  WDDM  |   00000000:01:00.0  On |
N/A |
|  0%   47C    P8             13W /  320W |    8051MiB /  16376MiB |
7%      Default |
|                                         |                        |
N/A |
+-----------------------------------------+------------------------
+---------------------+


+----------------------------------------------------------------------
--------------------+
| Processes:
|
|  GPU   GI   CI        PID   Type   Process name
GPU Memory |
|        ID   ID
Usage      |
|
======================================================================
===================|
|    0   N/A  N/A      2132    C+G   ...oogle\Chrome\Application\
chrome.exe      N/A      |
|    0   N/A  N/A      2480    C+G   ...ekyb3d8bbwe\
PhoneExperienceHost.exe      N/A      |
|    0   N/A  N/A      4948    C+G   ...US\ArmouryDevice\
asus_framework.exe      N/A      |
|    0   N/A  N/A      7340    C+G   ...CBS_cw5n1h2txyewy\
TextInputHost.exe      N/A      |
|    0   N/A  N/A      9700    C+G   ...crosoft\Edge\Application\
msedge.exe      N/A      |
|    0   N/A  N/A     11284    C+G   ...t.LockApp_cw5n1h2txyewy\
LockApp.exe      N/A      |
|    0   N/A  N/A     14016    C+G   ...siveControlPanel\
SystemSettings.exe      N/A      |
|    0   N/A  N/A     17532      C   ...er\anaconda3\envs\env_dm\
python.exe      N/A      |
|    0   N/A  N/A     18320    C+G   ...oogle\Chrome\Application\
chrome.exe      N/A      |
```

```
|    0   N/A  N/A     19044    C+G   ...nt.CBS_cw5n1h2txyewy\
SearchHost.exe      N/A      |
|    0   N/A  N/A     19196    C+G   ...5n1h2txyewy\
ShellExperienceHost.exe      N/A      |
|    0   N/A  N/A     19520    C+G   ...Programs\Microsoft VS Code\
Code.exe       N/A      |
|    0   N/A  N/A     21292    C+G   ...x64__qmba6cd70vzyy\
ArmouryCrate.exe       N/A      |
|    0   N/A  N/A     21768    C+G   C:\Windows\explorer.exe
N/A      |
|    0   N/A  N/A     22292    C+G   ...cal\Microsoft\OneDrive\
OneDrive.exe      N/A      |
|    0   N/A  N/A     24276    C+G   ...2txyewy\
StartMenuExperienceHost.exe      N/A      |
|    0   N/A  N/A     29272    C+G   ...on\131.0.2903.63\
msedgewebview2.exe      N/A      |
+----------------------------------------------------------------------
---------------------+
```

Data

Loading data emmotion and corpus marge to training and testing dataset

```python
import pandas as pd

input_path = 'data'
tweet_id = pd.read_csv(f'{input_path}/data_identification.csv')
tweet_id.head()
```

```
   tweet_id identification
0  0x28cc61           test
1  0x29e452          train
2  0x2b3819          train
3  0x2db41f           test
4  0x2a2acc          train
```

```python
train_tweet_id = tweet_id[tweet_id['identification'] ==
'train'].drop(['identification'], axis=1)
test_tweet_id = tweet_id[tweet_id['identification'] ==
'test'].drop(['identification'], axis=1)

train_tweet_id.head()
```

```
   tweet_id
1  0x29e452
2  0x2b3819
4  0x2a2acc
5  0x2a8830
6  0x20b21d
```

```
emotion_labels = pd.read_csv(f'{input_path}/emotion.csv')
emotion_labels.head()
```

```
   tweet_id       emotion
0  0x3140b1       sadness
1  0x368b73       disgust
2  0x296183  anticipation
3  0x2bd6e1           joy
4  0x2ee1dd  anticipation
```

```
tweets_df = pd.read_json(f'{input_path}/tweets_DM.json', lines=True)
source_df = pd.json_normalize(tweets_df['_source'])
tweets_df = pd.concat([tweets_df.drop(columns=['_source']),
source_df], axis=1)
tweets_df.head()
```

```
   _score           _index            _crawldate   _type  \
0     391  hashtag_tweets  2015-05-23 11:42:47  tweets
1     433  hashtag_tweets  2016-01-28 04:52:09  tweets
2     232  hashtag_tweets  2017-12-25 04:39:20  tweets
3     376  hashtag_tweets  2016-01-24 23:53:05  tweets
4     989  hashtag_tweets  2016-01-08 17:18:59  tweets

                  tweet.hashtags tweet.tweet_id  \
0                     [Snapchat]       0x376b20
1  [freepress, TrumpLegacy, CNN]       0x2d5350
2                   [bibleverse]       0x28b412
3                             []       0x1cd5b0
4                             []       0x2de201

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

```
# merge enmotion attribute
train_data = pd.merge(train_tweet_id, emotion_labels, on='tweet_id',
how='inner')
train_data = pd.merge(train_data, tweets_df, left_on='tweet_id',
right_on='tweet.tweet_id', how='inner')
train_data.head()
```

```
   tweet_id       emotion  _score           _index           _crawldate
\
0  0x29e452           joy     809  hashtag_tweets  2015-01-17 03:07:03

1  0x2b3819           joy     808  hashtag_tweets  2016-07-02 09:34:06

2  0x2a2acc         trust      16  hashtag_tweets  2016-08-15 18:18:39
```

```
3  0x2a8830          joy     768  hashtag_tweets  2017-02-11 08:49:46

4  0x20b21d  anticipation      70  hashtag_tweets  2016-11-23 05:37:10


    _type                                     tweet.hashtags
tweet.tweet_id  \
0  tweets                                                 []
0x29e452
1  tweets                               [spateradio, app]
0x2b3819
2  tweets                                                 []
0x2a2acc
3  tweets  [PUBG, GamersUnite, twitch, BeHealthy, StayPos...
0x2a8830
4  tweets                          [strength, bones, God]
0x20b21d

                                          tweet.text
0  Huge Respect @JohnnyVegasReal talking about l...
1  Yoooo we hit all our monthly goals with the ne...
2  @KIDSNTS @PICU_BCH @uhbcomms @BWCHBoss Well do...
3  Come join @ambushman27 on #PUBG while he striv...
4  @fanshixieen2014 Blessings!My #strength little...
```

```python
test_data = pd.merge(test_tweet_id, tweets_df, left_on='tweet_id',
right_on='tweet.tweet_id', how='inner')
test_data.head()
```

```
   tweet_id  _score          _index            _crawldate   _type  \
0  0x28cc61     107  hashtag_tweets  2017-01-17 14:13:32  tweets
1  0x2db41f     728  hashtag_tweets  2015-10-17 06:46:20  tweets
2  0x2466f6     491  hashtag_tweets  2016-12-19 03:50:27  tweets
3  0x23f9e9      28  hashtag_tweets  2017-04-09 19:32:19  tweets
4  0x1fb4e1     925  hashtag_tweets  2016-01-15 11:59:31  tweets

     tweet.hashtags tweet.tweet_id  \
0                []       0x28cc61
1                []       0x2db41f
2    [womendrivers]       0x2466f6
3  [robbingmembers]       0x23f9e9
4                []       0x1fb4e1

                                          tweet.text
0  @Habbo I've seen two separate colours of the e...
1  @FoxNews @KellyannePolls No serious self respe...
2  Looking for a new car, and it says 1 lady owne...
3  @cineworld “only the brave” just out and fount...
4  Felt like total dog 💩 going into open gym and ...
```

Data preprocessing

Preparing data for feeding into a model by, cleaning, transforming, and reduces noise in dataset. While the step are critical for ensuring data quility, their impact on model performance might be limited if there is unsufficient understanding of the specific model's requirement and its sensitivity to different types of feature. Maybe there tweet courpus is shot and clear enough, doesn't need additional pre-processing.

```
import re

def preprocess_text(text):
    # 移除<LH> 標籤
    text = text.replace('<LH>', '')

    # # 移除網址
    # text = re.sub(r'http\S+|www\S+', '', text)

    # # 移除提及標記(@用戶名)
    # text = re.sub(r'@\w+', '', text)

    # # 移除特殊字符或多餘的符號
    # text = re.sub(r'[^A-Za-z0-9\s.,!?]', '', text)

    # 壓縮多餘的空格
    text = re.sub(r'\s+', ' ', text).strip()

    return text

# Apply preprocessing function to the text column
train_data['processed_text'] =
train_data['tweet.text'].apply(preprocess_text)

train_data['tweet.text'].head()
```

```
0    Huge Respect□ @JohnnyVegasReal talking about l...
1    Yoooo we hit all our monthly goals with the ne...
2    @KIDSNTS @PICU_BCH @uhbcomms @BWCHBoss Well do...
3    Come join @ambushman27 on #PUBG while he striv...
4    @fanshixieen2014 Blessings!My #strength little...
Name: tweet.text, dtype: object
```

```
train_data['processed_text'].head()
```

```
0    Huge Respect□ @JohnnyVegasReal talking about l...
1    Yoooo we hit all our monthly goals with the ne...
2    @KIDSNTS @PICU_BCH @uhbcomms @BWCHBoss Well do...
3    Come join @ambushman27 on #PUBG while he striv...
4    @fanshixieen2014 Blessings!My #strength little...
Name: processed_text, dtype: object
```

```
from sklearn.model_selection import train_test_split

X = train_data['processed_text']
y = train_data['emotion']

X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2,
random_state=42)
```

Model training

This process involve tokenizing text into token IDs and attention mask, encoding lables, and batching using **data loaders**. The BERTweet is a large-scale pre-trained language model for English Tweets, the paper shows that the model producing better performance result on three Tweet NLP tasks: part-of-speech tagging, named-entity recognition and text classification, compare agaings previous state-of-the-art model RoBERTa-base and XLM-R-base with strong baseline. So, is reasonable to use BERTweet as the model for Tweets sentiment classification task.

```
import torch
from sklearn.preprocessing import LabelEncoder
from torch.utils.data import DataLoader, Dataset
from tqdm import tqdm
from sklearn.preprocessing import LabelEncoder
from transformers import BertweetTokenizer,
AutoModelForSequenceClassification
from torch.optim.lr_scheduler import StepLR
from torch.optim import AdamW

def encode_data(X_train, X_val, y_train, y_val, tokenizer,
max_length=128):
    # Tokenize text data
    train_encodings = tokenizer(list(X_train), truncation=True,
padding=True, max_length=max_length, return_tensors="pt")
    val_encodings = tokenizer(list(X_val), truncation=True,
padding=True, max_length=max_length, return_tensors="pt")

    # Encode labels
    label_encoder = LabelEncoder()
    train_labels = label_encoder.fit_transform(y_train)
    val_labels = label_encoder.transform(y_val)

    return train_encodings, val_encodings, train_labels, val_labels,
label_encoder

# Dataset class for loading text data from DataFrame
class TweetDataset(Dataset):
    def __init__(self, encodings, labels):
        self.encodings = encodings  # Should already contain tensors
if return_tensors="pt" was used
```

```
        self.labels = torch.tensor(labels)  # Ensure labels are
converted to a tensor

    def __getitem__(self, idx):
        # Use encodings as they are without re-wrapping in
torch.tensor
        item = {key: val[idx] for key, val in self.encodings.items()}
        item["labels"] = self.labels[idx]
        return item

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

from sklearn.metrics import classification_report

def train_model(model, optimizer, scheduler, train_dataloader,
val_dataloader, device, num_epochs, label_encoder,
save_dir="./models"):
    for epoch in range(num_epochs):
        print(f"\nEpoch {epoch + 1}/{num_epochs}")
        model.train()
        total_train_loss = 0

        # Training phase
        for batch in tqdm(train_dataloader, desc="Training"):
            b_input_ids = batch['input_ids'].to(device)
            b_attention_mask = batch['attention_mask'].to(device)
            b_labels = batch['labels'].to(device).long()

            # Zero the gradients
            optimizer.zero_grad()

            # Forward pass
            outputs = model(input_ids=b_input_ids,
attention_mask=b_attention_mask, labels=b_labels)
            loss = outputs.loss
            total_train_loss += loss.item()

            # Backward pass
            loss.backward()

            # Clip the gradient to prevent exploding gradients
            torch.nn.utils.clip_grad_norm_(model.parameters(),
max_norm=1.0)

            # Update parameters
            optimizer.step()

        # Step the scheduler after each epoch
        scheduler.step()
```

```
        avg_train_loss = total_train_loss / len(train_dataloader)
        print(f"Epoch {epoch + 1}/{num_epochs}, Training Loss:
{avg_train_loss}")

        # Validation phase
        model.eval()
        total_val_loss = 0
        all_preds = []
        all_labels = []

        with torch.no_grad():
            for batch in tqdm(val_dataloader, desc="Validating"):
                b_input_ids = batch['input_ids'].to(device)
                b_attention_mask = batch['attention_mask'].to(device)
                b_labels = batch['labels'].to(device).long()

                # Forward pass for validation
                outputs = model(input_ids=b_input_ids,
attention_mask=b_attention_mask, labels=b_labels)
                loss = outputs.loss
                logits = outputs.logits

                total_val_loss += loss.item()

                # Store predictions and true labels
                _, preds = torch.max(logits, dim=1)
                all_preds.extend(preds.cpu().numpy())
                all_labels.extend(b_labels.cpu().numpy())

        avg_val_loss = total_val_loss / len(val_dataloader)

        # Decode numerical labels to original class names
        decoded_preds = label_encoder.inverse_transform(all_preds)
        decoded_labels = label_encoder.inverse_transform(all_labels)

        # Generate classification report
        class_report = classification_report(decoded_labels,
decoded_preds, digits=4, target_names=label_encoder.classes_)

        print(f"Epoch {epoch + 1}/{num_epochs}, Validation Loss:
{avg_val_loss}")
        print("\nClassification Report:")
        print(class_report)

        # Save model checkpoint
        save_path = f"{save_dir}/ep_{epoch + 1}"
        model.save_pretrained(save_path)
        print(f"Model checkpoint saved to {save_path}")
```

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(device)
```

```
cuda
```

```
model_name = "vinai/bertweet-base"
tokenizer = BertweetTokenizer.from_pretrained(model_name,
normalization=True)
model = AutoModelForSequenceClassification.from_pretrained(model_name,
num_labels=8).to(device)
```

```
Some weights of RobertaForSequenceClassification were not initialized
from the model checkpoint at vinai/bertweet-base and are newly
initialized: ['classifier.dense.bias', 'classifier.dense.weight',
'classifier.out_proj.bias', 'classifier.out_proj.weight']
You should probably TRAIN this model on a down-stream task to be able
to use it for predictions and inference.
```

```
X_train_encoding, X_val_encoding, y_train_label, y_val_label,
label_encoder = encode_data(X_train, X_val, y_train, y_val, tokenizer,
max_length=48)

# Initialize datasets
train_dataset = TweetDataset(X_train_encoding, y_train_label)
val_dataset = TweetDataset(X_val_encoding, y_val_label)

train_dataloader = DataLoader(train_dataset, batch_size=64,
shuffle=True)
val_dataloader = DataLoader(val_dataset, batch_size=64)

optimizer = AdamW(model.parameters(), lr=2e-5)
scheduler = StepLR(optimizer, step_size=2, gamma=0.7)
train_model(model, optimizer, scheduler, train_dataloader,
val_dataloader, device, num_epochs=1, label_encoder=label_encoder,
save_dir="./models/Bertweet_v3")
```

```
Epoch 1/1

Training: 100%|██████████| 18195/18195 [27:19<00:00, 11.10it/s]

Epoch 1/1, Training Loss: 1.0013337626336662

Validating: 100%|██████████| 4549/4549 [01:54<00:00, 39.87it/s]

Epoch 1/1, Validation Loss: 0.918088096488987

Classification Report:
              precision    recall  f1-score   support

       anger     0.5513    0.4216    0.4778      7964
anticipation     0.7191    0.7447    0.7317     49725
```

```
     disgust     0.5363    0.5633    0.5495     27892
        fear     0.7345    0.5895    0.6541     12955
         joy     0.7265    0.8062    0.7643    103089
     sadness     0.5301    0.6925    0.6005     38835
    surprise     0.6945    0.3253    0.4431      9750
       trust     0.7473    0.4410    0.5547     40903

    accuracy                         0.6697    291113
   macro avg     0.6550    0.5730    0.5970    291113
weighted avg     0.6782    0.6697    0.6633    291113

Model checkpoint saved to ./models/Bertweet_v3/ep_1
```

INFERENCE

Sentiment classification task involves tokenizing the input text using the BERTweet tokenizer, converting it into token IDs and attention masks, and passing these inputs into the pre-trained BERTweet model. During the forward pass, the token IDs are transformed into dense embeddings by the model's embedding layer, which are then processed through the transformer layers to generate logits representing class probabilities and futher inference emotions.

```
from transformers import BertweetTokenizer,
AutoModelForSequenceClassification
import torch
from tqdm import tqdm
from sklearn.preprocessing import LabelEncoder

test_data['processed_text'] =
test_data['tweet.text'].apply(preprocess_text)
X_test = test_data['processed_text']

# Load the tokenizer and model
model_name = "vinai/bertweet-base"
tokenizer = BertweetTokenizer.from_pretrained(model_name,
normalization=True)
test_encodings = tokenizer(list(X_test), truncation=True,
padding=True, max_length=48, return_tensors="pt")

# Create a DataLoader for batching
test_dataset = torch.utils.data.TensorDataset(
    test_encodings["input_ids"],
    test_encodings["attention_mask"]
)
test_dataloader =
torch.utils.data.DataLoader(test_dataset,batch_size=512)

model_path = "./models/Bertweet/"
model =
```

```
AutoModelForSequenceClassification.from_pretrained(model_path).to(devi
ce)

# Set the model to evaluation mode
model.eval()

# Perform predictions
predictions = []
with torch.no_grad():
    for batch in tqdm(test_dataloader, desc="Testing"):
        input_ids = batch[0].to(device)
        attention_mask = batch[1].to(device)

        # Get logits from the model
        outputs = model(input_ids=input_ids,
attention_mask=attention_mask)
        logits = outputs.logits

        # Get the predicted labels
        preds = torch.argmax(logits, dim=1)
        predictions.extend(preds.cpu().numpy())

# Map numerical predictions to text labels
label_encoder = LabelEncoder()
label_encoder.fit(y_train)  # Ensure this matches your training labels
predicted_labels = label_encoder.inverse_transform(predictions)

# Add predictions to the test DataFrame
test_data['emotion'] = predicted_labels

# Keep only the desired columns
result = test_data[['tweet_id', 'emotion']]

result = result.rename(columns={'tweet_id': 'id'})
result.to_csv("submission.csv", index=False)

Testing: 100%|██████████| 805/805 [02:29<00:00,  5.37it/s]
```

Submission trials

Follow the recommendations provided in the example to remove time-related information. These messages are unrelated to sentiment and may be considered noise during model training. Attempt to remove usernames, excessive symbols, and spaces. Based on the results of public submissions, removing usernames, excessive symbols, and spaces reduces accuracy. This indicates that too much information is being removed. The results show that only removing time-related information is sufficient, leaving the rest to be handled by the language model.

Reference

1.https://arxiv.org/abs/2005.10200
2.https://huggingface.co/docs/transformers/model_doc/bertweet
3.https://www.kaggle.com/code/gauravgupta9158/twitter-sentiment-analysis