Install necessary libraries and packages

```
!pip install torch
import torch
import pandas as pd
from tqdm.notebook import tqdm
```

```
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packages (2.3.0+cu121)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch) (3.15.4)
Requirement already satisfied: typing-extensions>=4.8.0 in /usr/local/lib/python3.10/dist-packages (from torch) (4.12.2)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch) (1.12.1)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch) (3.3)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch) (3.1.4)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from torch) (2023.6.0)
Collecting nvidia-cuda-nvrtc-cu12==12.1.105 (from torch)
  Using cached nvidia_cuda_nvrtc_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (23.7 MB)
Collecting nvidia-cuda-runtime-cu12==12.1.105 (from torch)
  Using cached nvidia_cuda_runtime_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (823 kB)
Collecting nvidia-cuda-cupti-cu12==12.1.105 (from torch)
  Using cached nvidia_cuda_cupti_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (14.1 MB)
Collecting nvidia-cudnn-cu12==8.9.2.26 (from torch)
  Using cached nvidia_cudnn_cu12-8.9.2.26-py3-none-manylinux1_x86_64.whl (731.7 MB)
Collecting nvidia-cublas-cu12==12.1.3.1 (from torch)
  Using cached nvidia_cublas_cu12-12.1.3.1-py3-none-manylinux1_x86_64.whl (410.6 MB)
Collecting nvidia-cufft-cu12==11.0.2.54 (from torch)
  Using cached nvidia_cufft_cu12-11.0.2.54-py3-none-manylinux1_x86_64.whl (121.6 MB)
Collecting nvidia-curand-cu12==10.3.2.106 (from torch)
  Using cached nvidia_curand_cu12-10.3.2.106-py3-none-manylinux1_x86_64.whl (56.5 MB)
Collecting nvidia-cusolver-cu12==11.4.5.107 (from torch)
  Using cached nvidia_cusolver_cu12-11.4.5.107-py3-none-manylinux1_x86_64.whl (124.2 MB)
Collecting nvidia-cusparse-cu12==12.1.0.106 (from torch)
  Using cached nvidia_cusparse_cu12-12.1.0.106-py3-none-manylinux1_x86_64.whl (196.0 MB)
Collecting nvidia-nccl-cu12==2.20.5 (from torch)
  Using cached nvidia_nccl_cu12-2.20.5-py3-none-manylinux2014_x86_64.whl (176.2 MB)
Collecting nvidia-nvtx-cu12==12.1.105 (from torch)
  Using cached nvidia_nvtx_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (99 kB)
Requirement already satisfied: triton==2.3.0 in /usr/local/lib/python3.10/dist-packages (from torch) (2.3.0)
Collecting nvidia-nvjitlink-cu12 (from nvidia-cusolver-cu12==11.4.5.107->torch)
  Downloading nvidia_nvjitlink_cu12-12.5.82-py3-none-manylinux2014_x86_64.whl (21.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21.3/21.3 MB 59.2 MB/s eta 0:00:00
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->torch) (2.1.5)
Requirement already satisfied: mpmath<1.4.0,>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from sympy->torch) (1.3.0)
Installing collected packages: nvidia-nvtx-cu12, nvidia-nvjitlink-cu12, nvidia-nccl-cu12, nvidia-curand-cu12, nvidia-cufft-cu12, nvidia-
Successfully installed nvidia-cublas-cu12-12.1.3.1 nvidia-cuda-cupti-cu12-12.1.105 nvidia-cuda-nvrtc-cu12-12.1.105 nvidia-cuda-runtime-c
```

Import the data set

```
import pandas as pd
input_df = pd.read_csv('/content/Twitter_Data.csv')
```

Only 20% of Data used for Training and Inference

```
df = input_df.sample(frac=0.2, random_state=42)
```

```
df
```

	clean_text	category
90053	modi full too drama package with habbit eating...	1.0
44179	better show that “entire political science” de...	1.0
9998	you must say what you witnessed since 2014 you...	1.0
57234	who are they expect that janpath will congratu...	0.0
81150	bjp policy only divide and rule bjp propogand...	0.0
...	...	...
155202	you are the one that hates modi	0.0
86762	this the case why doesn’ modi announce prohibi...	-1.0
110509	can narendra modi placed leo now	0.0
72942	terror attacks have not killed zero civilians ...	-1.0
28367	rajiv kumar there was hurdle when your governm...	1.0

31088 rows × 2 columns

Find the Null value

```
df.isnull().sum()
```

```
clean_text    0
category      0
label         0
data_type     0
dtype: int64
```

Drop the null value

```
df.dropna(inplace=True)
```

check after the Drop the null

```
df.isnull().sum()
```

```
clean_text    0
category      0
dtype: int64
```

Convert the label from folat to Integer

```
df['category'] = df['category'].astype(int)
```

```
df
```

	clean_text	category
22511	even the world' top economies fails provide 10...	-1
102942	vahiyat opinion hai tax bracket its okay for r...	1
127087	sir you will find insane lovers modi almost ne...	-1
15661	may loser but why dont you become winner telli...	0
64921	yadav jee your hate modi attitude has made you...	-1
...	...	...
123613	vajpayee has started the gst and modi has impl...	1
76395	mention abuses that congress has showered modi...	0
100806	modi' stinky farts eagerly lapped the lap dog...	0
16757	\nindia launched 104 satellites one and now ai...	0
53658	modi funny comment for vijay mallya national y...	1

26266 rows × 2 columns

Unique Value of label

```
unique_values = df['category'].unique()
```

```
unique_values
```

```
array([ 1,  0, -1])
```

Now Map the label to string

```
mapping = {
    -1: 'negative',
    0: 'neutral',
    1: 'positive'
}

# Apply the mapping
df['category'] = df['category'].map(mapping)
```

```
df
```

	clean_text	category
90053	modi full too drama package with habbit eating...	positive
44179	better show that "entire political science" de...	positive
9998	you must say what you witnessed since 2014 you...	positive
57234	who are they expect that janpath will congratu...	neutral
81150	bjp policy only divide and rule bjp propogand...	neutral
...	...	...
155202	you are the one that hates modi	neutral
86762	this the case why doesn' modi announce prohibi...	negative
110509	can narendra modi placed leo now	neutral
72942	terror attacks have not killed zero civilians ...	negative
28367	rajiv kumar there was hurdle when your governm...	positive

31087 rows × 2 columns

```
df.columns
```

```
Index(['clean_text', 'category'], dtype='object')
```

```
#info
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 31087 entries, 90053 to 28367
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   clean_text  31087 non-null  object
 1   category    31087 non-null  object
dtypes: object(2)
memory usage: 728.6+ KB
```

```
df.clean_text.iloc[10]
```

```
'that ques why did isro chairman drdo director not announce that  bcoz than modi wah wa
h nahi hoti '
```

count number no label

```
df.category.value_counts()
```

```
category
positive    13818
neutral     10517
negative     6752
Name: count, dtype: int64
```

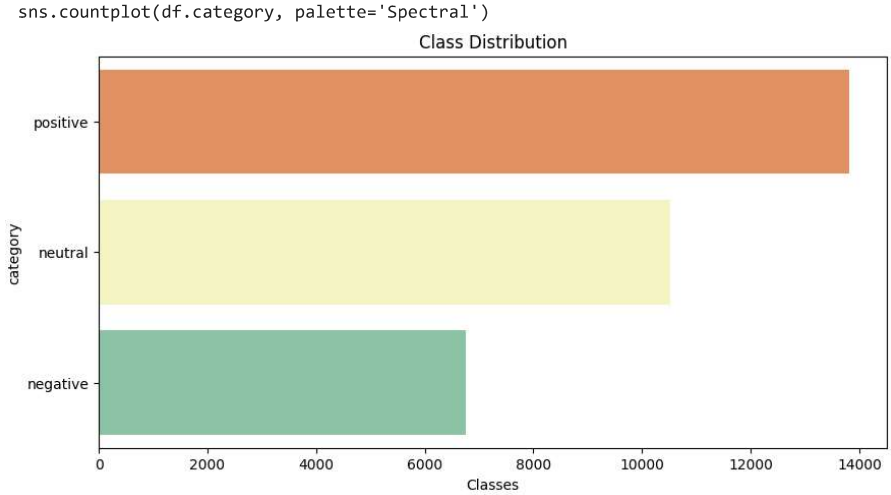
```
import matplotlib.pyplot as plt
import seaborn as sns

#plot class distribution
plt.figure(figsize=(10, 5))
sns.countplot(df.category, palette='Spectral')
plt.xlabel('Classes')
plt.title('Class Distribution');
```

```
<ipython-input-30-9bd44cc90bb4>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.

  sns.countplot(df.category, palette='Spectral')
```




```
#store classes into an array
possible_labels = df.category.unique()
possible_labels
```

```
array(['positive', 'neutral', 'negative'], dtype=object)
```

```
#convert labels into numeric values
label_dict = {}
for index, possible_label in enumerate(possible_labels):
    label_dict[possible_label] = index
```

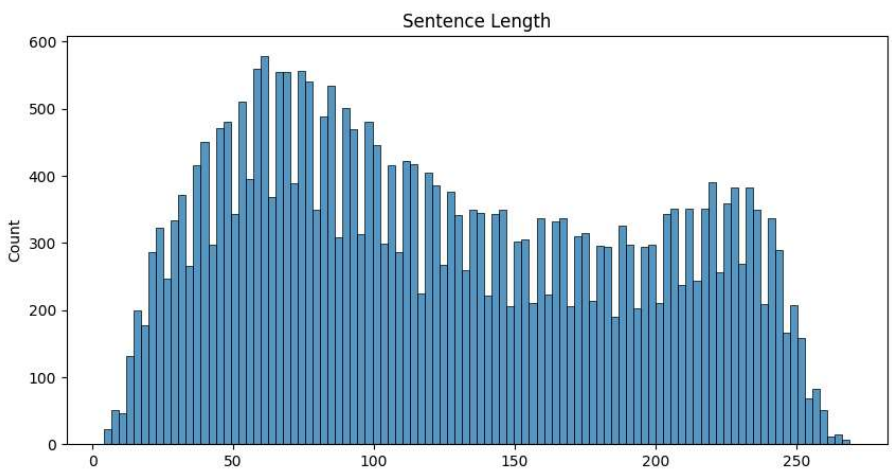
```
label_dict
```

```
{'positive': 0, 'neutral': 1, 'negative': 2}
```

```
#convert labels into numeric values
df['label'] = df.category.replace(label_dict)
df.head(10)
```

	clean_text	category	label
90053	modi full too drama package with habbit eating...	positive	0
44179	better show that “entire political science” de...	positive	0
9998	you must say what you witnessed since 2014 you...	positive	0
57234	who are they expect that janpath will congratu...	neutral	1
81150	bjp policy only divide and rule bjp propogand...	neutral	1
134586	support modi for better future india	positive	0
14041	new research from analyzed more than 9000 twee...	positive	0
59499	modi can defeated only modi decides not contes...	positive	0
50485	jee narendra modi quite elder and senior thn y...	neutral	1
140346	lutyens delhi better prepare the boy aka youth...	positive	0

```
#need equal length sentences
#plot hist of sentence length
plt.figure(figsize=(10, 5))
sns.histplot([len(s) for s in df.clean_text], bins=100)
plt.title('Sentence Length')
plt.show()
```



find the maximum length In clean Text

```
max_len = max([len(sent) for sent in df.clean_text])
print('Max length: ', max_len)
```

```
Max length:  269
```

```
from sklearn.model_selection import train_test_split

#train test split
X_train, X_val, y_train, y_val = train_test_split(df.index.values,
                                                  df.label.values,
                                                  test_size = 0.15,
                                                  random_state = 17,
                                                  stratify = df.label.values)
```

create new column

```
df['data_type'] = ['not_set'] * df.shape[0]
df.head()
```

	clean_text	category	label	data_type
90053	modi full too drama package with habbit eating...	positive	0	not_set
44179	better show that “entire political science” de...	positive	0	not_set
9998	you must say what you witnessed since 2014 you...	positive	0	not_set
57234	who are they expect that janpath will congratu...	neutral	1	not_set
81150	bjp policy only divide and rule bjp propogand...	neutral	1	not_set

```
#fill in data type
df.loc[X_train, 'data_type'] = 'train'
df.loc[X_val, 'data_type'] = 'val'
```

```
df.groupby(['category', 'label', 'data_type']).count()
```

			clean_text
category	**label**	**data_type**	
negative	**2**	**train**	5739
		val	1013
neutral	**1**	**train**	8939
		val	1578
positive	**0**	**train**	11745
		val	2073

```
!pip install transformers

from transformers import BertTokenizer
from torch.utils.data import TensorDataset
```

```
Requirement already satisfied: transformers in /usr/local/lib/python3.10/dist-packages (4.41.2)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from transformers) (3.15.4)
Requirement already satisfied: huggingface-hub<1.0,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.23.4)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from transformers) (1.25.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from transformers) (24.1)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from transformers) (6.0.1)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-packages (from transformers) (2024.5.15)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from transformers) (2.31.0)
Requirement already satisfied: tokenizers<0.20,>=0.19 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.19.1)
Requirement already satisfied: safetensors>=0.4.1 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.4.3)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-packages (from transformers) (4.66.4)
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.23.0->transform
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.23.0-
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (2024.6.2)
```

load tokenizer

```
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased',
                                          do_lower_case = True)
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:89: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https:/
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public model
  warnings.warn(
tokenizer_config.json: 100%  48.0/48.0 [00:00<00:00, 3.74kB/s]
vocab.txt: 100%  232k/232k [00:00<00:00, 3.59MB/s]
tokenizer.json: 100%  466k/466k [00:00<00:00, 2.35MB/s]
/usr/local/lib/python3.10/dist-packages/huggingface_hub/file_download.py:1132: FutureWar
  warnings.warn(
config.json: 100%  570/570 [00:00<00:00, 41.3kB/s]
```

tokenize train set

```
encoded_data_train = tokenizer.batch_encode_plus(df[df.data_type == 'train'].clean_text.values,
                                                 add_special_tokens = True,
                                                 return_attention_mask = True,
                                                 pad_to_max_length = True,
                                                 max_length = 270,
                                                 return_tensors = 'pt')
```

```
Truncation was not explicitly activated but `max_length` is provided a specific value, please use `truncation=True` to explicitly trunca
/usr/local/lib/python3.10/dist-packages/transformers/tokenization_utils_base.py:2699: FutureWarning: The `pad_to_max_length` argument is
  warnings.warn(
```

tokenizer val set

```
encoded_data_val = tokenizer.batch_encode_plus(df[df.data_type == 'val'].clean_text.values,
                                               #add_special_tokens = True,
                                               return_attention_mask = True,
                                               pad_to_max_length = True,
                                               max_length = 270,
                                               return_tensors = 'pt')
```

```
encoded_data_train
```

```
{'input_ids': tensor([[  101, 16913,  2072,  ...,     0,     0,     0],
        [  101,  2040,  2024,  ...,     0,     0,     0],
        [  101, 24954,  3343,  ...,     0,     0,     0],
        ...,
        [  101,  2064,  6583,  ...,     0,     0,     0],
        [  101,  7404,  4491,  ...,     0,     0,     0],
        [  101, 11948, 12848,  ...,     0,     0,     0]]), 'token_type_ids': tensor([[0, 0, 0,  ..., 0, 0, 0],
        [0, 0, 0,  ..., 0, 0, 0],
        [0, 0, 0,  ..., 0, 0, 0],
        ...,
        [0, 0, 0,  ..., 0, 0, 0],
        [0, 0, 0,  ..., 0, 0, 0],
        [0, 0, 0,  ..., 0, 0, 0]]), 'attention_mask': tensor([[1, 1, 1,  ..., 0, 0, 0],
        [1, 1, 1,  ..., 0, 0, 0],
        [1, 1, 1,  ..., 0, 0, 0],
        ...,
        [1, 1, 1,  ..., 0, 0, 0],
        [1, 1, 1,  ..., 0, 0, 0],
        [1, 1, 1,  ..., 0, 0, 0]])}
```

Used for encode train set

```
input_ids_train = encoded_data_train['input_ids']
attention_masks_train = encoded_data_train['attention_mask']
labels_train = torch.tensor(df[df.data_type == 'train'].label.values)
```

used for encode val set

```
input_ids_val = encoded_data_val['input_ids']
attention_masks_val = encoded_data_val['attention_mask']
#convert data type to torch.tensor
labels_val = torch.tensor(df[df.data_type == 'val'].label.values)
```

```
input_ids_train
```

```
tensor([[  101, 16913,  2072,  ...,     0,     0,     0],
        [  101,  2040,  2024,  ...,     0,     0,     0],
        [  101, 24954,  3343,  ...,     0,     0,     0],
        ...,
        [  101,  2064,  6583,  ...,     0,     0,     0],
        [  101,  7404,  4491,  ...,     0,     0,     0],
        [  101, 11948, 12848,  ...,     0,     0,     0]])
```

```
attention_masks_train
```

```
tensor([[1, 1, 1,  ..., 0, 0, 0],
        [1, 1, 1,  ..., 0, 0, 0],
        [1, 1, 1,  ..., 0, 0, 0],
        ...,
        [1, 1, 1,  ..., 0, 0, 0],
        [1, 1, 1,  ..., 0, 0, 0],
        [1, 1, 1,  ..., 0, 0, 0]])
```

```
labels_train
```

```
tensor([0, 1, 1,  ..., 1, 2, 0])
```

create dataloader

```
dataset_train = TensorDataset(input_ids_train,
                              attention_masks_train,
                              labels_train)

dataset_val = TensorDataset(input_ids_val,
                            attention_masks_val,
                            labels_val)
```

```
print(len(dataset_train))
print(len(dataset_val))
```

```
26423
4664
```

```
dataset_train
```

```
<torch.utils.data.dataset.TensorDataset at 0x7edacb8afd90>
```

```
dataset_train.tensors
```

```
(tensor([[  101, 16913,  2072,  ...,     0,     0,     0],
         [  101,  2040,  2024,  ...,     0,     0,     0],
         [  101, 24954,  3343,  ...,     0,     0,     0],
         ...,
         [  101,  2064,  6583,  ...,     0,     0,     0],
         [  101,  7404,  4491,  ...,     0,     0,     0],
         [  101, 11948, 12848,  ...,     0,     0,     0]]),
 tensor([[1, 1, 1,  ..., 0, 0, 0],
         [1, 1, 1,  ..., 0, 0, 0],
         [1, 1, 1,  ..., 0, 0, 0],
         ...,
         [1, 1, 1,  ..., 0, 0, 0],
         [1, 1, 1,  ..., 0, 0, 0],
         [1, 1, 1,  ..., 0, 0, 0]]),
 tensor([0, 1, 1,  ..., 1, 2, 0]))
```

load pre-trained BERT bert-base-uncased

```
from transformers import BertForSequenceClassification
```

```
model = BertForSequenceClassification.from_pretrained('bert-base-uncased'
```

```
model = BertForSequenceClassification.from_pretrained( bert-base-uncased ,
                                                       num_labels = len(label_dict),
                                                       output_attentions = False,
                                                       output_hidden_states = False)
```

```
model.safetensors: 100%                                    440M/440M [00:01<00:00, 300MB/s]
Some weights of BertForSequenceClassification were not initialized from the model checkpoint at bert-base-uncased and are newly initiali
```

model summary

```
model.config
```

```
BertConfig {
  "_name_or_path": "bert-base-uncased",
  "architectures": [
    "BertForMaskedLM"
  ],
  "attention_probs_dropout_prob": 0.1,
  "classifier_dropout": null,
  "gradient_checkpointing": false,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "id2label": {
    "0": "LABEL_0",
    "1": "LABEL_1",
    "2": "LABEL_2"
  },
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "label2id": {
    "LABEL_0": 0,
    "LABEL_1": 1,
    "LABEL_2": 2
  },
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "pad_token_id": 0,
  "position_embedding_type": "absolute",
  "transformers_version": "4.41.2",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 30522
}
```

```
from torch.utils.data import DataLoader, RandomSampler, SequentialSampler

batch_size = 4
dataloader_train = DataLoader(dataset_train,sampler = RandomSampler(dataset_train),batch_size = batch_size)
dataloader_val = DataLoader(dataset_val,sampler = RandomSampler(dataset_val),batch_size = 32)
```

```
from transformers import AdamW, get_linear_schedule_with_warmup
epochs = 10

optimizer = AdamW(model.parameters(),
                  lr = 1e-5,
                  eps = 1e-8)
```

```
/usr/local/lib/python3.10/dist-packages/transformers/optimization.py:588: FutureWarning: This implementation of AdamW is deprecated and
  warnings.warn(
```

```
scheduler = get_linear_schedule_with_warmup(optimizer,num_warmup_steps = 0,num_training_steps = len(dataloader_train)*epochs)
```

```
import numpy as np
from sklearn.metrics import f1_score

def f1_score_func(preds, labels):
    preds_flat = np.argmax(preds, axis=1).flatten()
    labels_flat = labels.flatten()
    return f1_score(labels_flat, preds_flat, average = 'weighted')
```

```
def accuracy_per_class(preds, labels):
    label_dict_inverse = {v: k for k, v in label_dict.items()}

    preds_flat = np.argmax(preds, axis=1).flatten()
    labels_flat = labels.flatten()

    for label in np.unique(labels_flat):
        y_preds = preds_flat[labels_flat==label]
        y_true = labels_flat[labels_flat==label]
        print(f'Class: {label_dict_inverse[label]}')
        print(f'Accuracy:{len(y_preds[y_preds==label])}/{len(y_true)}\n')


def evaluate(dataloader_val):
    model.eval()

    loss_val_total = 0
    predictions, true_vals = [], []

    for batch in tqdm(dataloader_val):
        batch = tuple(b.to(device) for b in batch)

        inputs = {'input_ids':      batch[0],
                  'attention_mask': batch[1],
                  'labels':         batch[2]}
        with torch.no_grad():
            outputs = model(**inputs)
        loss = outputs[0]
        logits = outputs[1]
        loss_val_total += loss.item()
        logits = logits.detach().cpu().numpy()
        label_ids = inputs['labels'].cpu().numpy()
        predictions.append(logits)
        true_vals.append(label_ids)
    loss_val_avg = loss_val_total/len(dataloader_val)
    predictions = np.concatenate(predictions, axis=0)
    true_vals = np.concatenate(true_vals, axis=0)

    return loss_val_avg, predictions, true_vals


import random

seed_val = 17
random.seed(seed_val)
np.random.seed(seed_val)
torch.manual_seed(seed_val)
torch.cuda.manual_seed_all(seed_val)


device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model.to(device)
print(device)
```

```
cuda
```

```
for epoch in tqdm(range(1, epochs+1)):
    model.train()
    loss_train_total = 0
    progress_bar = tqdm(dataloader_train,
                        desc='Epoch {:1d}'.format(epoch),
                        leave=False,
                        disable=False)

    for batch in progress_bar:
        model.zero_grad()
        batch = tuple(b.to(device) for b in batch)
        inputs = {'input_ids': batch[0],
                  'attention_mask': batch[1],
                  'labels': batch[2]}

        outputs = model(**inputs)
        loss = outputs[0]
        loss_train_total +=loss.item()
        loss.backward()
        torch.nn.utils.clip_grad_norm_(model.parameters(), 1.0)
        optimizer.step()
        scheduler.step()

        progress_bar.set_postfix({'training_loss': '{:.3f}'.format(loss.item()/len(batch))})

    tqdm.write('\nEpoch {epoch}')
    loss_train_avg = loss_train_total/len(dataloader_train)
    tqdm.write(f'Training loss: {loss_train_avg}')

    val_loss, predictions, true_vals = evaluate(dataloader_val)
    val_f1 = f1_score_func(predictions, true_vals)
    tqdm.write(f'Validation loss: {val_loss}')
    tqdm.write(f'F1 Score (weighted): {val_f1}')
```

```
100%                                          10/10  [2:00:27<00:00,  722.91s/it]


Epoch {epoch}
Training loss: 0.5495109339078306
100%                                          146/146  [00:35<00:00,   4.55it/s]
Validation loss: 0.3921652176490768
F1 Score (weighted): 0.9207897344635848


Epoch {epoch}
Training loss: 0.266881305562972
100%                                          146/146  [00:35<00:00,   4.55it/s]
Validation loss: 0.2583814689344152
F1 Score (weighted): 0.9474626429749421
```

```
outputs.loss
```

```
tensor(1.0053e-05, device='cuda:0', grad_fn=<NllLossBackward0>)
```

```
outputs.logits
```

```
tensor([[-4.3464, -3.5174,  7.9818],
        [ 8.2082, -3.8661, -4.4356],
        [ 8.3571, -3.9621, -4.6562]], device='cuda:0',
       grad_fn=<AddmmBackward0>)
Training loss: 0.10209534198475068
```

```
#save model
model.to(device)
pass


#evaluate
_, predictions, true_vals = evaluate(dataloader_val)
```

```
100%                                          146/146  [00:35<00:00,   4.53it/s]
```

```
accuracy_per_class(predictions, true_vals)
```

```
Class: positive
Accuracy:2006/2073

Class: neutral
Accuracy:1537/1578

Class: negative
Accuracy:950/1013
```

```
from sklearn.metrics import precision_score, recall_score, confusion_matrix
model.eval()
predictions, true_vals = [], []

for batch in dataloader_val:
    batch = tuple(b.to(device) for b in batch)

    with torch.no_grad():
        inputs = {'input_ids': batch[0],
                  'attention_mask': batch[1],
                  'labels': batch[2]}
        outputs = model(**inputs)

    logits = outputs[1]
```