```
!pip install --quiet wandb
!pip install --quiet transformers
!pip install --quiet datasets
!pip install --quiet emoji
!pip install --quiet kaggle
!pip install --quiet torchinfo
!pip install --quiet imbalanced-learn
!pip install --quiet gdown
!pip install --quiet clean-text
!pip install --quiet accelerate -U
!pip install --quiet transformers[torch]
!pip install --quiet huggingface_hub[tensorflow]
!pip install --quiet huggingface_hub[cli,torch]
!pip install --quiet openai

from huggingface_hub import HfApi
import huggingface_hub

#transformers
import transformers
from transformers import AutoTokenizer
from transformers import GPT2Tokenizer
from transformers import BertTokenizer
from transformers import OpenAIGPTTokenizer
from transformers import AutoConfig
from transformers import GPT2Config
from transformers.models.bert.modeling_bert import BertModel
from transformers.models.gpt2.modeling_gpt2 import GPT2Model
from transformers.models.openai.modeling_openai import OpenAIGPTModel
from transformers.models.bert.modeling_bert import BertPreTrainedModel
from transformers.models.gpt2.modeling_gpt2 import GPT2PreTrainedModel
from transformers.models.openai.modeling_openai import OpenAIGPTPreTrainedModel
from transformers import BertForSequenceClassification
from transformers import GPT2ForSequenceClassification
from transformers import OpenAIGPTForSequenceClassification
from transformers.modeling_outputs import SequenceClassifierOutput
from transformers import Trainer
from transformers import TrainingArguments
from transformers import get_scheduler
from transformers import TextDataset
from transformers.modeling_utils import PreTrainedModel

#torch
import torch
import torch.nn as nn
from torch.nn import BCEWithLogitsLoss, MSELoss, CrossEntropyLoss
from torch.optim import AdamW
from torch.utils import data
from torch.utils.data import Dataset as ds, DataLoader

# dataset
import datasets
from datasets import Dataset
from datasets import Sequence
from datasets import Value
from datasets import Features
from datasets import ClassLabel
from datasets import DatasetDict

# sklearn
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, f1_score
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import ConfusionMatrixDisplay, confusion_matrix

#open ai
import openai

# others
import os
import wandb
import re, string
import emoji
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import pickle
import joblib
import traceback
from tqdm import tqdm
from collections import defaultdict

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

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.1/2.1 MB 29.1 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 188.6/188.6 kB 23.3 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 218.8/218.8 kB 24.5 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 62.7/62.7 kB 7.0 MB/s eta 0:00:00
  Building wheel for pathtools (setup.py) ... done
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.6/7.6 MB 24.3 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 268.8/268.8 kB 27.0 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.8/7.8 MB 42.9 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 37.5 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 519.6/519.6 kB 2.0 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 115.3/115.3 kB 14.8 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 194.1/194.1 kB 24.8 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 134.8/134.8 kB 18.1 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 358.9/358.9 kB 7.0 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 175.4/175.4 kB 2.2 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 53.1/53.1 kB 4.2 MB/s eta 0:00:00
  Building wheel for emoji (setup.py) ... done
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 251.2/251.2 kB 3.4 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 67.7/67.7 kB 1.6 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 76.5/76.5 kB 1.3 MB/s eta 0:00:00
```

```
! huggingface-cli login --token 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
```

```
Token will not been saved to git credential helper. Pass `add_to_git_credential=True` if you want to set the git credential as well.
Token is valid (permission: write).
Your token has been saved to /root/.cache/huggingface/token
Login successful
```

```
huggingface_hub.login()
```

Copy a token from your Hugging Face tokens page and paste it below.

Immediately click login after copying your token or it might be stored in plain text in this notebook file.

Token:

☑ Add token as git credential?

Login

Pro Tip: If you don't already have one, you can create a dedicated 'notebooks' token with

```
class BertForClassification(BertForSequenceClassification):

    def __init__(self, config):
        super().__init__(config)
        self.num_labels = config.num_labels
        self.config = config

        # Load model body > return all og the HS
        self.bert = BertModel(config)
        # Set up token classification head
        self.dropout = nn.Dropout(config.hidden_dropout_prob)
        self.classifier = nn.Linear(config.hidden_size, config.num_labels)

        # Initialize weights and apply final processing
        self.post_init()


    def forward(self, input_ids=None, attention_mask=None, token_type_ids=None,
                labels=None, **kwargs):
        # Use model body to get encoder representations
        outputs = self.bert(input_ids, attention_mask=attention_mask,
                            token_type_ids=token_type_ids, **kwargs)

        # Apply classifier to encoder representation > [cls]
        sequence_output = self.dropout(outputs[1])
        logits = self.classifier(sequence_output)

        # Calculate losses
        loss = None
        if labels is not None:
            loss_fct = nn.CrossEntropyLoss()
            loss = loss_fct(logits.view(-1, self.num_labels), labels.view(-1))
            #outputs = (loss,) + outputs # can comment

        # return outputs  # (loss), logits, (hidden_states), (attentions)


        # Return model output object
        return SequenceClassifierOutput(
            loss=loss,
            logits=logits,
            hidden_states=outputs.hidden_states,
            attentions=outputs.attentions,
        )


model_names = ["bert-large-uncased"]

id2label = {0: 'non-toxic', 1: 'mild', 2: 'toxic'}
label2id = { v:k for (k,v) in id2label.items()}
labels = ['non-toxic','mild', 'toxic']

tokenizers = {"bert-large-uncased": BertTokenizer.from_pretrained("bert-large-uncased")}
configs = {"bert-large-uncased": AutoConfig.from_pretrained("bert-large-uncased", num_labels=3, id2label=id2label, label2id=label2id)}
models = {"bert-large-uncased": BertForClassification.from_pretrained("bert-large-uncased", config=configs["bert-large-uncased"])}
```

```
Downloading (…)solve/main/vocab.txt: 100%                    232k/232k [00:00<00:00, 9.22MB/s]

Downloading (…)okenizer_config.json: 100%                    28.0/28.0 [00:00<00:00, 1.96kB/s]

Downloading (…)lve/main/config.json: 100%                    571/571 [00:00<00:00, 34.9kB/s]

Downloading model.safetensors: 100%                          1.34G/1.34G [00:05<00:00, 146MB/s]

Some weights of BertForClassification were not initialized from the model checkpoint at bert-large-uncased and are newly initialized: ['
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
```

```
def clean_text(text):
  text = re.sub('#', '', text) # Removing '#' hashtag
  text = re.sub('\w+:\/\/\S+', '', text) # Removing hyperlink
  text = re.sub('[^a-zA-Z]', ' ', text) # Remove punctuation
  return text.lower()

def compute_metrics(pred):
  labels = pred.label_ids
  preds = pred.predictions.argmax(-1)
  f1 = f1_score(labels, preds, average="weighted")
  acc = accuracy_score(labels, preds)
  return {"accuracy": acc, "f1": f1}

def create_dataset(df, text, label):
  class_names = ['non-toxic','mild', 'toxic']
  data_dict = {'text':df[text], 'labels':df[label]}
  tags = ClassLabel(num_classes=3 , names=class_names)
  feature_set = Features({'text':Value(dtype='string'), 'labels':tags})

  return Dataset.from_dict(mapping = data_dict, features = feature_set)


def get_dataset():
  train = pd.read_csv('/content/train.csv')
  validation = pd.read_csv('/content/validation.csv')

  dataset_train = create_dataset(train,"message","target")
  dataset_val = create_dataset(validation,"message","target")

  train_convert = train.copy()
  train_convert['target'] = train_convert['target'].astype(str)
  train_convert['message'] = (train_convert['message'] + ' ->')
  train_convert['target'] = (' ' + train_convert['target'])
  train_convert.rename(columns={'message':'prompt', 'target':'completion'}, inplace=True)
  train_convert.to_json("train_pandas.jsonl", orient='records', lines=True)

  validation_convert = validation.copy()
  validation_convert['target'] = validation_convert['target'].astype(str)
  validation_convert['message'] = (validation_convert['message'] + ' ->')
  validation_convert['target'] = (' ' + validation_convert['target'])
  validation_convert.rename(columns={'message':'prompt', 'target':'completion'}, inplace=True)
  validation_convert.to_json("validation_pandas.jsonl", orient='records', lines=True)

  dataset = DatasetDict()
  dataset["train"] = dataset_train
  dataset["validation"] = dataset_val
  return dataset
```

```
! wandb login eed5796f17cdf020038ca37377edaaec8c9ddfbe
```

```
wandb: Appending key for api.wandb.ai to your netrc file: /root/.netrc
```

```
wandb.login()
wandb.init(project="bert-toxic-chat-detection", entity="dani-squad")
```

```
wandb: Currently logged in as: dffesalbon (dani-squad). Use `wandb login --relogin` to force relogin
Tracking run with wandb version 0.15.10
Run data is saved locally in /content/wandb/run-20230907_001043-rj53tmy9
Syncing run fancy-blaze-32 to Weights & Biases (docs)
View project at https://wandb.ai/dani-squad/bert-toxic-chat-detection
View run at https://wandb.ai/dani-squad/bert-toxic-chat-detection/runs/rj53tmy9
Display W&B run
```

```
def finetune_bert(model_name, dataset, max_len=64, train_batch_size=16, num_epochs=5):

  tokenizer = tokenizers[model_name]
  model = models[model_name]

  tokenize = lambda batch: tokenizer(batch["text"], padding=True, truncation=True, max_length=max_len)

  encoded = dataset.map(tokenize, batched=True, batch_size=32)

  model.to(device)
  optimizer = AdamW(model.parameters(), lr=2e-5, eps=1e-8)
  logging_steps = len(encoded['train']) // train_batch_size
  num_training_steps = num_epochs * logging_steps

  scheduler = get_scheduler(
      name="linear",
      optimizer=optimizer,
      num_warmup_steps=0,
      num_training_steps=num_training_steps)

  training_args = TrainingArguments(
      output_dir=f"{model_name}-toxic-detector",
      num_train_epochs=num_epochs,
      per_device_train_batch_size=train_batch_size,
      per_device_eval_batch_size=train_batch_size,
      weight_decay=0.01,
      evaluation_strategy="epoch",
      save_steps=1e6,
      disable_tqdm=False,
      logging_steps=logging_steps,
      push_to_hub=False,
      log_level="error",
      report_to="wandb",
      run_name=model_name)

  optimizers = (optimizer, scheduler)

  model trainer = Trainer(
```

```
def save_model(model_name, model_trainer):
  model_trainer.save_model(f'/content/{model_name}-saved')
  api.upload_folder(
      folder_path=f"/content/{model_name}-saved",
      repo_id=f"dffesalbon/{model_name}-dota-toxic",
      repo_type="model")
```

```
dataset = get_dataset()
  model.eval()
```

BERT (Large-uncased)

```
model_name, model_trainer, encoded = finetune_bert(model_names[0], dataset)
```

```
Map: 100%                                   1722/1722 [00:00<00:00, 2067.87 examples/s]
Map: 100%                                   192/192 [00:00<00:00, 1157.71 examples/s]
                                            [540/540 04:01, Epoch 5/5]
```

Epoch	Training Loss	Validation Loss	Accuracy	F1
1	0.857000	0.568056	0.786458	0.781023
2	0.555000	0.417159	0.838542	0.837147
3	0.384100	0.504736	0.796875	0.788389
4	0.231400	0.592420	0.807292	0.803610
5	0.141100	0.615863	0.817708	0.819774

```
try:
  save_model(model_name, model_trainer)
except Exception as ex:
```