```
!pip install --quiet datasets
!pip install --quiet emoji
!pip install --quiet torchinfo
!pip install --quiet imbalanced-learn
!pip install --quiet gdown
!pip install --quiet clean-text
!pip install --quiet accelerate -U
```

```
    File "/usr/local/lib/python3.10/dist-packages/pip/_internal/operations/check.py", line 42, in create_package_set_from_installed
      dependencies = list(dist.iter_dependencies())
    File "/usr/local/lib/python3.10/dist-packages/pip/_internal/metadata/pkg_resources.py", line 216, in iter_dependencies
      return self._dist.requires(extras)
    File "/usr/local/lib/python3.10/dist-packages/pip/_vendor/pkg_resources/__init__.py", line 2821, in requires
      dm = self._dep_map
    File "/usr/local/lib/python3.10/dist-packages/pip/_vendor/pkg_resources/__init__.py", line 3110, in _dep_map
      self.__dep_map = self._compute_dependencies()
    File "/usr/local/lib/python3.10/dist-packages/pip/_vendor/pkg_resources/__init__.py", line 3120, in _compute_dependencies
      reqs.extend(parse_requirements(req))
    File "/usr/local/lib/python3.10/dist-packages/pip/_vendor/pkg_resources/__init__.py", line 3173, in __init__
      super(Requirement, self).__init__(requirement_string)
    File "/usr/local/lib/python3.10/dist-packages/pip/_vendor/packaging/requirements.py", line 102, in __init__
      req = REQUIREMENT.parseString(requirement_string)
    File "/usr/local/lib/python3.10/dist-packages/pip/_vendor/pyparsing/core.py", line 1131, in parse_string
      loc, tokens = self._parse(instring, 0)
    File "/usr/local/lib/python3.10/dist-packages/pip/_vendor/pyparsing/core.py", line 817, in _parseNoCache
      loc, tokens = self.parseImpl(instring, pre_loc, doActions)
    File "/usr/local/lib/python3.10/dist-packages/pip/_vendor/pyparsing/core.py", line 3886, in parseImpl
      loc, exprtokens = e._parse(instring, loc, doActions)
    File "/usr/local/lib/python3.10/dist-packages/pip/_vendor/pyparsing/core.py", line 817, in _parseNoCache
      loc, tokens = self.parseImpl(instring, pre_loc, doActions)
    File "/usr/local/lib/python3.10/dist-packages/pip/_vendor/pyparsing/core.py", line 4114, in parseImpl
      return e._parse(
    File "/usr/local/lib/python3.10/dist-packages/pip/_vendor/pyparsing/core.py", line 817, in _parseNoCache
      loc, tokens = self.parseImpl(instring, pre_loc, doActions)
    File "/usr/local/lib/python3.10/dist-packages/pip/_vendor/pyparsing/core.py", line 3864, in parseImpl
      loc, resultlist = self.exprs[0]._parse(
    File "/usr/local/lib/python3.10/dist-packages/pip/_vendor/pyparsing/core.py", line 817, in _parseNoCache
      loc, tokens = self.parseImpl(instring, pre_loc, doActions)
    File "/usr/local/lib/python3.10/dist-packages/pip/_vendor/pyparsing/core.py", line 3864, in parseImpl
      loc, resultlist = self.exprs[0]._parse(
    File "/usr/local/lib/python3.10/dist-packages/pip/_vendor/pyparsing/core.py", line 776, in _parseNoCache
      def _parseNoCache(
  KeyboardInterrupt

  During handling of the above exception, another exception occurred:

  Traceback (most recent call last):
    File "/usr/lib/python3.10/logging/__init__.py", line 1732, in isEnabledFor
      return self._cache[level]
  KeyError: 50

  During handling of the above exception, another exception occurred:

  Traceback (most recent call last):
    File "/usr/local/bin/pip3", line 8, in <module>
      sys.exit(main())
    File "/usr/local/lib/python3.10/dist-packages/pip/_internal/cli/main.py", line 79, in main
      return command.main(cmd_args)
    File "/usr/local/lib/python3.10/dist-packages/pip/_internal/cli/base_command.py", line 101, in main
      return self._main(args)
    File "/usr/local/lib/python3.10/dist-packages/pip/_internal/cli/base_command.py", line 223, in _main
      return run(options, args)
    File "/usr/local/lib/python3.10/dist-packages/pip/_internal/cli/base_command.py", line 206, in exc_logging_wrapper
      logger.critical("Operation cancelled by user")
    File "/usr/lib/python3.10/logging/__init__.py", line 1523, in critical
      if self.isEnabledFor(CRITICAL):
    File "/usr/lib/python3.10/logging/__init__.py", line 1732, in isEnabledFor
```

```
#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

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


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', 'severe']
  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 split_dataset(random_seed=42):
  np.random.seed(random_seed)
  torch.manual_seed(random_seed)
  data = pd.read_csv('/content/trim.csv')
  other_fields = ['date time', 'match', 'region', 'game time', 'slot', 'player slot']
  data = data.drop(other_fields, 1)
  class_map = {'non-toxic': 0, 'mild': 1, 'severe': 2}
  data['target'] = data.apply(lambda x: class_map[x['toxicity']], axis=1)
  data = data.drop('toxicity', 1)

  #split data, get 25% for testing
  train, test = train_test_split(data, test_size = 0.25, random_state = 0)
  #clean the data
  train.loc[:,'message'] = train['message'].apply(lambda x : clean_text(x))
  test.loc[:,'message'] = test['message'].apply(lambda x : clean_text(x))
  #split training data, get 10% for validation
  train_all_convert = train.copy()
  train_all_convert['target'] = train_all_convert['target'].astype(str)
  train_all_convert.rename(columns={'message':'prompt', 'target':'completion'}, inplace=True)
  train_all_convert.to_json("train_all_pandas.jsonl", orient='records', lines=True)

  train, validation = train_test_split(train, test_size=0.1, random_state=random_seed, stratify=train['target'])
  train.to_csv('/content/train.csv', index=False)
  validation.to_csv('/content/validation.csv', index=False)
  test.to_csv('/content/test.csv', index=False)




split_dataset()
```

```
    <ipython-input-7-c5ecc2a10ff9>:6: FutureWarning: In a future version of pandas all arguments of DataFrame.drop except for the argument
      data = data.drop(other_fields, 1)
```

```
<ipython-input-7-c5ecc2a10ff9>:9: FutureWarning: In a future version of pandas all arguments of DataFrame.drop except for the argument
  data = data.drop('toxicity', 1)
```

✓ 0s completed at 8:07 PM