```
!pip install datasets
```

```
Requirement already satisfied: datasets in c:\users\gjk\anaconda3\lib\
site-packages (2.15.0)
Requirement already satisfied: dill<0.3.8,>=0.3.0 in c:\users\gjk\
anaconda3\lib\site-packages (from datasets) (0.3.7)
Requirement already satisfied: pandas in c:\users\gjk\anaconda3\lib\
site-packages (from datasets) (1.4.4)
Requirement already satisfied: pyarrow-hotfix in c:\users\gjk\
anaconda3\lib\site-packages (from datasets) (0.6)
Requirement already satisfied: numpy>=1.17 in c:\users\gjk\anaconda3\
lib\site-packages (from datasets) (1.23.5)
Requirement already satisfied: aiohttp in c:\users\gjk\anaconda3\lib\
site-packages (from datasets) (3.8.1)
Requirement already satisfied: fsspec[http]<=2023.10.0,>=2023.1.0 in
c:\users\gjk\anaconda3\lib\site-packages (from datasets) (2023.10.0)
Requirement already satisfied: huggingface-hub>=0.18.0 in c:\users\
gjk\anaconda3\lib\site-packages (from datasets) (0.19.4)
Requirement already satisfied: requests>=2.19.0 in c:\users\gjk\
anaconda3\lib\site-packages (from datasets) (2.28.1)
Requirement already satisfied: packaging in c:\users\gjk\anaconda3\
lib\site-packages (from datasets) (22.0)
Requirement already satisfied: xxhash in c:\users\gjk\anaconda3\lib\
site-packages (from datasets) (3.4.1)
Requirement already satisfied: pyarrow>=8.0.0 in c:\users\gjk\
anaconda3\lib\site-packages (from datasets) (14.0.1)
Requirement already satisfied: multiprocess in c:\users\gjk\anaconda3\
lib\site-packages (from datasets) (0.70.15)
Requirement already satisfied: pyyaml>=5.1 in c:\users\gjk\anaconda3\
lib\site-packages (from datasets) (6.0)
Requirement already satisfied: tqdm>=4.62.1 in c:\users\gjk\anaconda3\
lib\site-packages (from datasets) (4.64.1)
Requirement already satisfied: frozenlist>=1.1.1 in c:\users\gjk\
anaconda3\lib\site-packages (from aiohttp->datasets) (1.2.0)
Requirement already satisfied: multidict<7.0,>=4.5 in c:\users\gjk\
anaconda3\lib\site-packages (from aiohttp->datasets) (5.1.0)
Requirement already satisfied: attrs>=17.3.0 in c:\users\gjk\
anaconda3\lib\site-packages (from aiohttp->datasets) (22.1.0)
Requirement already satisfied: async-timeout<5.0,>=4.0.0a3 in c:\
users\gjk\anaconda3\lib\site-packages (from aiohttp->datasets) (4.0.1)
Requirement already satisfied: yarl<2.0,>=1.0 in c:\users\gjk\
anaconda3\lib\site-packages (from aiohttp->datasets) (1.6.3)
Requirement already satisfied: aiosignal>=1.1.2 in c:\users\gjk\
anaconda3\lib\site-packages (from aiohttp->datasets) (1.2.0)
Requirement already satisfied: charset-normalizer<3.0,>=2.0 in c:\
users\gjk\anaconda3\lib\site-packages (from aiohttp->datasets) (2.0.4)
Requirement already satisfied: typing-extensions>=3.7.4.3 in c:\users\
gjk\anaconda3\lib\site-packages (from huggingface-hub>=0.18.0-
>datasets) (4.7.1)
Requirement already satisfied: filelock in c:\users\gjk\anaconda3\lib\
site-packages (from huggingface-hub>=0.18.0->datasets) (3.9.0)
Requirement already satisfied: idna<4,>=2.5 in c:\users\gjk\anaconda3\
lib\site-packages (from requests>=2.19.0->datasets) (3.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\gjk\
anaconda3\lib\site-packages (from requests>=2.19.0->datasets)
(1.26.14)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\gjk\
anaconda3\lib\site-packages (from requests>=2.19.0->datasets)
(2022.12.7)
Requirement already satisfied: colorama in c:\users\gjk\anaconda3\lib\
site-packages (from tqdm>=4.62.1->datasets) (0.4.6)
Requirement already satisfied: pytz>=2020.1 in c:\users\gjk\anaconda3\
lib\site-packages (from pandas->datasets) (2022.7)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\gjk\
anaconda3\lib\site-packages (from pandas->datasets) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\users\gjk\anaconda3\lib\
site-packages (from python-dateutil>=2.8.1->pandas->datasets) (1.16.0)

WARNING: Ignoring invalid distribution -rotobuf (c:\users\gjk\appdata\
roaming\python\python39\site-packages)
WARNING: Ignoring invalid distribution - (c:\users\gjk\appdata\
roaming\python\python39\site-packages)
WARNING: Ignoring invalid distribution -ensorflow-intel (c:\users\gjk\
appdata\roaming\python\python39\site-packages)
WARNING: Ignoring invalid distribution -orch (c:\users\gjk\anaconda3\
lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\gjk\appdata\
roaming\python\python39\site-packages)
WARNING: Ignoring invalid distribution - (c:\users\gjk\appdata\
roaming\python\python39\site-packages)
WARNING: Ignoring invalid distribution -ensorflow-intel (c:\users\gjk\
appdata\roaming\python\python39\site-packages)
WARNING: Ignoring invalid distribution -orch (c:\users\gjk\anaconda3\
lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\gjk\appdata\
roaming\python\python39\site-packages)
WARNING: Ignoring invalid distribution - (c:\users\gjk\appdata\
roaming\python\python39\site-packages)
WARNING: Ignoring invalid distribution -ensorflow-intel (c:\users\gjk\
appdata\roaming\python\python39\site-packages)
WARNING: Ignoring invalid distribution -orch (c:\users\gjk\anaconda3\
lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\gjk\appdata\
roaming\python\python39\site-packages)
WARNING: Ignoring invalid distribution - (c:\users\gjk\appdata\
roaming\python\python39\site-packages)
WARNING: Ignoring invalid distribution -ensorflow-intel (c:\users\gjk\
appdata\roaming\python\python39\site-packages)
WARNING: Ignoring invalid distribution -orch (c:\users\gjk\anaconda3\
lib\site-packages)
```

```
WARNING: Ignoring invalid distribution -rotobuf (c:\users\gjk\appdata\
roaming\python\python39\site-packages)
WARNING: Ignoring invalid distribution - (c:\users\gjk\appdata\
roaming\python\python39\site-packages)
WARNING: Ignoring invalid distribution -ensorflow-intel (c:\users\gjk\
appdata\roaming\python\python39\site-packages)
WARNING: Ignoring invalid distribution -orch (c:\users\gjk\anaconda3\
lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\gjk\appdata\
roaming\python\python39\site-packages)
WARNING: Ignoring invalid distribution - (c:\users\gjk\appdata\
roaming\python\python39\site-packages)
WARNING: Ignoring invalid distribution -ensorflow-intel (c:\users\gjk\
appdata\roaming\python\python39\site-packages)
WARNING: Ignoring invalid distribution -orch (c:\users\gjk\anaconda3\
lib\site-packages)
```

```
from datasets import load_dataset
import torch
from torch import nn
from torch.utils.data import DataLoader, Dataset
from datasets import load_dataset
from torchtext.data.utils import get_tokenizer
from torchtext.vocab import build_vocab_from_iterator
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import string
from torch.utils.data import random_split
import numpy as np

from transformers import BertTokenizer, BertForSequenceClassification,
AdamW
from transformers import get_linear_schedule_with_warmup

device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

label_columns = ['related', 'PII', 'request', 'offer', 'aid_related',
'medical_help',
                 'medical_products', 'search_and_rescue', 'security',
'military',
                 'child_alone', 'water', 'food', 'shelter',
'clothing', 'money',
                 'missing_people', 'refugees', 'death', 'other_aid',
'infrastructure_related',
                  'transport', 'buildings', 'electricity', 'tools',
'hospitals', 'shops',
                 'aid_centers', 'other_infrastructure',
'weather_related', 'floods', 'storm',
                 'fire', 'earthquake', 'cold', 'other_weather',
'direct_report']
num_labels = len(label_columns)

tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')

def preprocess_for_bert(data):
    # Tokenize the messages and prepare the labels
    input_ids = []
    attention_masks = []
    labels = []

    for i in range(len(data)):
        encoded = tokenizer.encode_plus(
            data[i]['message'],
            add_special_tokens=True,
            max_length=128,  # Adjust as needed
            padding='max_length',
            truncation=True,
            return_attention_mask=True,
            return_tensors='pt',
        )
        input_ids.append(encoded['input_ids'])
        attention_masks.append(encoded['attention_mask'])
        labels.append([data[i][label] for label in label_columns])

    input_ids = torch.cat(input_ids, dim=0)
    attention_masks = torch.cat(attention_masks, dim=0)
    labels = torch.tensor(labels, dtype=torch.float)

    return input_ids, attention_masks, labels
dataset = load_dataset("disaster_response_messages")
# Apply preprocessing
train_input_ids, train_attention_masks, train_labels =
preprocess_for_bert(dataset['train'])
val_input_ids, val_attention_masks, val_labels =
preprocess_for_bert(dataset['validation'])
test_input_ids, test_attention_masks, test_labels =
preprocess_for_bert(dataset['test'])

from torch.utils.data import TensorDataset

batch_size = 16

# Create the DataLoader for our training set
train_data = TensorDataset(train_input_ids, train_attention_masks,
train_labels)
train_sampler = torch.utils.data.RandomSampler(train_data)
train_dataloader = DataLoader(train_data, sampler=train_sampler,
batch_size=batch_size)
```

```
# Create the DataLoader for our validation set
validation_data = TensorDataset(val_input_ids, val_attention_masks,
val_labels)
validation_sampler =
torch.utils.data.SequentialSampler(validation_data)
validation_dataloader = DataLoader(validation_data,
sampler=validation_sampler, batch_size=batch_size)

test_data = TensorDataset(test_input_ids, test_attention_masks,
test_labels)
test_sampler = torch.utils.data.SequentialSampler(test_data)
test_dataloader = DataLoader(test_data, sampler=test_sampler,
batch_size=batch_size)

model = BertForSequenceClassification.from_pretrained('bert-base-
uncased', num_labels=num_labels)
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model.to(device)
```

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

```
BertForSequenceClassification(
  (bert): BertModel(
    (embeddings): BertEmbeddings(
      (word_embeddings): Embedding(30522, 768, padding_idx=0)
      (position_embeddings): Embedding(512, 768)
      (token_type_embeddings): Embedding(2, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-12,
elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (encoder): BertEncoder(
      (layer): ModuleList(
        (0-11): 12 x BertLayer(
          (attention): BertAttention(
            (self): BertSelfAttention(
              (query): Linear(in_features=768, out_features=768,
bias=True)
              (key): Linear(in_features=768, out_features=768,
bias=True)
              (value): Linear(in_features=768, out_features=768,
bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): BertSelfOutput(
              (dense): Linear(in_features=768, out_features=768,
bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-12,
elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): BertIntermediate(
            (dense): Linear(in_features=768, out_features=3072,
bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): BertOutput(
            (dense): Linear(in_features=3072, out_features=768,
bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12,
elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
      )
    )
    (pooler): BertPooler(
      (dense): Linear(in_features=768, out_features=768, bias=True)
      (activation): Tanh()
    )
  )
  (dropout): Dropout(p=0.1, inplace=False)
  (classifier): Linear(in_features=768, out_features=37, bias=True)
)
```

```
optimizer = AdamW(model.parameters(), lr=5e-5)
epochs = 50  # Adjust as needed
total_steps = len(train_dataloader) * epochs

scheduler = get_linear_schedule_with_warmup(optimizer,
                                            num_warmup_steps=0,

num_training_steps=total_steps)
loss_fn = torch.nn.BCEWithLogitsLoss()
```

```
c:\Users\GJK\anaconda3\lib\site-packages\transformers\
optimization.py:411: FutureWarning: This implementation of AdamW is
deprecated and will be removed in a future version. Use the PyTorch
implementation torch.optim.AdamW instead, or set
`no_deprecation_warning=True` to disable this warning
  warnings.warn(
```

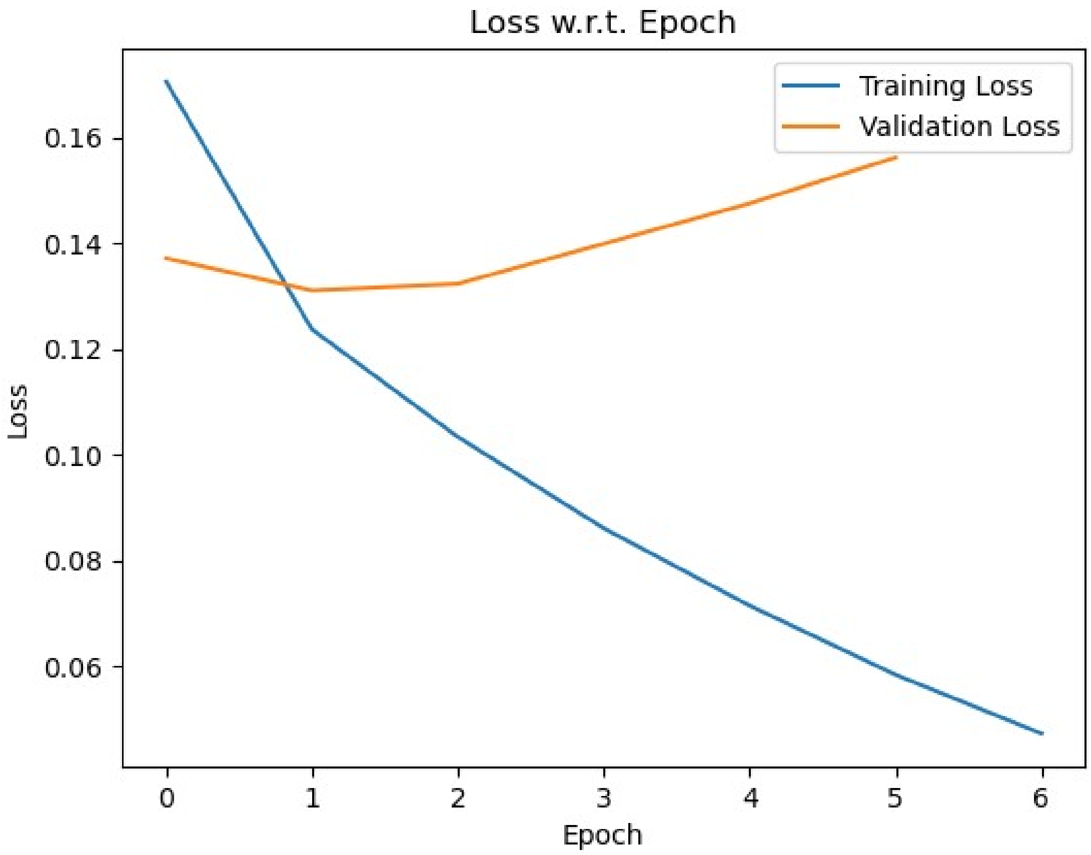
```
def multilabel_accuracy(preds, labels, threshold=0.5):
    preds = torch.sigmoid(preds)
    preds = (preds > threshold).float()
    correct = (preds == labels).float()
    acc = correct.sum() / correct.numel()
    return acc
    accuracy = multilabel_accuracy(outputs.logits, batch_labels)

def validate(model, dataloader, loss_fn, device):
    model.eval()
    total_loss, total_accuracy = 0, 0

    for batch in dataloader:
        inputs, attention_masks, labels = batch[0].to(device),
batch[1].to(device), batch[2].to(device)

        with torch.no_grad():
            outputs = model(input_ids=inputs,
attention_mask=attention_masks)
            loss = loss_fn(outputs.logits, labels)
            acc = multilabel_accuracy(outputs.logits, labels)

        total_loss += loss.item()
        total_accuracy += acc.item()

    avg_loss = total_loss / len(dataloader)
    avg_acc = total_accuracy / len(dataloader)
    return avg_loss, avg_acc



trainloss_history = []
validloss_history = []
for epoch in range(epochs):
    model.train()
    total_loss, total_accuracy = 0, 0

    for batch in train_dataloader:
        inputs, attention_masks, labels = batch[0].to(device),
batch[1].to(device), batch[2].to(device)
        optimizer.zero_grad()

        outputs = model(input_ids=inputs,
attention_mask=attention_masks)


        loss = loss_fn(outputs.logits, labels)
        loss.backward()
        optimizer.step()
        acc = multilabel_accuracy(outputs.logits, labels)

        total_loss += loss.item()
        total_accuracy += acc.item()
    avg_train_loss = total_loss / len(train_dataloader)
    avg_train_acc = total_accuracy / len(train_dataloader)
    if len(trainloss_history) > 5:
        largest = max(trainloss_history[-5:])
        if avg_train_loss > largest:
            break
    trainloss_history.append(avg_train_loss)




    print(f"Epoch {epoch + 1}/{epochs} - Loss: {avg_train_loss:.4f}, 
Accuracy: {avg_train_acc:.4f}")


    val_loss, val_accuracy = validate(model, validation_dataloader, 
loss_fn, device)
    print(f'Validation Loss: {val_loss}, Validation Accuracy: 
{val_accuracy}')
    if len(valiloss_history) > 5:
        largest = max(valiloss_history[-5:])
        if val_loss > largest:
            break
    valiloss_history.append(val_loss)
```

```
Epoch 1/50 - Loss: 0.1706, Accuracy: 0.9428
Validation Loss: 0.13713375583105947, Validation Accuracy: 
0.952776166593066
Epoch 2/50 - Loss: 0.1237, Accuracy: 0.9572
Validation Loss: 0.1310825933303152, Validation Accuracy: 
0.953613095031762
Epoch 3/50 - Loss: 0.1034, Accuracy: 0.9638
Validation Loss: 0.13232915767509004, Validation Accuracy: 
0.9541158979723913
Epoch 4/50 - Loss: 0.0861, Accuracy: 0.9694
Validation Loss: 0.13989342559347367, Validation Accuracy: 
0.9542054837534887
Epoch 5/50 - Loss: 0.0715, Accuracy: 0.9740
Validation Loss: 0.14751040644478053, Validation Accuracy: 
0.9540690887048378
Epoch 6/50 - Loss: 0.0584, Accuracy: 0.9780
Validation Loss: 0.15619565979780062, Validation Accuracy: 
0.9536857312510473
Epoch 7/50 - Loss: 0.0473, Accuracy: 0.9814
Validation Loss: 0.1656121688067728, Validation Accuracy: 
0.9526187907094541
```

```python
def test(model, dataloader, loss_fn, device):
    model.eval()
    total_loss, total_accuracy = 0, 0

    for batch in dataloader:
        inputs, attention_masks, labels = batch[0].to(device), 
batch[1].to(device), batch[2].to(device)

        with torch.no_grad():
            outputs = model(input_ids=inputs, 
attention_mask=attention_masks)
            loss = loss_fn(outputs.logits, labels)
            acc = multilabel_accuracy(outputs.logits, labels)

        total_loss += loss.item()
        total_accuracy += acc.item()

    avg_loss = total_loss / len(dataloader)
    avg_acc = total_accuracy / len(dataloader)
    return avg_loss, avg_acc

# Example usage after completing all training epochs
test_loss, test_accuracy = test(model, test_dataloader, loss_fn, 
device)
print(f'Test Loss: {test_loss}, Test Accuracy: {test_accuracy}')
```

```
Test Loss: 0.13558429052883927, Test Accuracy: 0.9620086240045952
```

```python
import matplotlib.pyplot as plt
import copy
TL = copy.deepcopy(trainloss_history)
VL = copy.deepcopy(valiloss_history)
L = [TL,VL]
fig0 = plt.figure(0)
for i, loss in enumerate(L):
    if i == 0:
        plt.plot(loss, label='Training Loss')
    if i == 1:
        plt.plot(loss, label='Validation Loss')

plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Loss w.r.t. Epoch')
plt.legend()
plt.show()
```



```
# torch.save(model.state_dict(), 'bert_model.pt')
```