BERT

```
!pip install datasets scikit-learn

import torch
import torch.nn as nn
import torch.nn.functional as F
from torch.optim import AdamW

import os
import numpy as np
from tqdm.auto import tqdm

import random
from torch.utils.data import DataLoader

from datasets import load_dataset
from transformers import (
    BertTokenizerFast,
    BertForSequenceClassification,
    DataCollatorWithPadding,
    get_linear_schedule_with_warmup,
)

RANDOM_SEED = 42
torch.manual_seed(RANDOM_SEED)
random.seed(RANDOM_SEED)

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

```
Collecting datasets
  Downloading datasets-3.5.1-py3-none-any.whl.metadata (19 kB)
Requirement already satisfied: scikit-learn in
/usr/local/lib/python3.11/dist-packages (1.6.1)
Requirement already satisfied: filelock in
/usr/local/lib/python3.11/dist-packages (from datasets) (3.18.0)
Requirement already satisfied: numpy>=1.17 in
/usr/local/lib/python3.11/dist-packages (from datasets) (2.0.2)
Requirement already satisfied: pyarrow>=15.0.0 in
/usr/local/lib/python3.11/dist-packages (from datasets) (18.1.0)
Collecting dill<0.3.9,>=0.3.0 (from datasets)
  Downloading dill-0.3.8-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: pandas in
/usr/local/lib/python3.11/dist-packages (from datasets) (2.2.2)
Requirement already satisfied: requests>=2.32.2 in
/usr/local/lib/python3.11/dist-packages (from datasets) (2.32.3)
Requirement already satisfied: tqdm>=4.66.3 in
/usr/local/lib/python3.11/dist-packages (from datasets) (4.67.1)
Collecting xxhash (from datasets)
  Downloading xxhash-3.5.0-cp311-cp311-
```

```
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (12 kB)
Collecting multiprocess<0.70.17 (from datasets)
  Downloading multiprocess-0.70.16-py311-none-any.whl.metadata (7.2
kB)
Collecting fsspec<=2025.3.0,>=2023.1.0 (from
fsspec[http]<=2025.3.0,>=2023.1.0->datasets)
  Downloading fsspec-2025.3.0-py3-none-any.whl.metadata (11 kB)
Requirement already satisfied: aiohttp in
/usr/local/lib/python3.11/dist-packages (from datasets) (3.11.15)
Requirement already satisfied: huggingface-hub>=0.24.0 in
/usr/local/lib/python3.11/dist-packages (from datasets) (0.30.2)
Requirement already satisfied: packaging in
/usr/local/lib/python3.11/dist-packages (from datasets) (24.2)
Requirement already satisfied: pyyaml>=5.1 in
/usr/local/lib/python3.11/dist-packages (from datasets) (6.0.2)
Requirement already satisfied: scipy>=1.6.0 in
/usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.15.2)
Requirement already satisfied: joblib>=1.2.0 in
/usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in
/usr/local/lib/python3.11/dist-packages (from scikit-learn) (3.6.0)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets)
(2.6.1)
Requirement already satisfied: aiosignal>=1.1.2 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets)
(1.3.2)
Requirement already satisfied: attrs>=17.3.0 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets)
(25.3.0)
Requirement already satisfied: frozenlist>=1.1.1 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets)
(1.6.0)
Requirement already satisfied: multidict<7.0,>=4.5 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets)
(6.4.3)
Requirement already satisfied: propcache>=0.2.0 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets)
(0.3.1)
Requirement already satisfied: yarl<2.0,>=1.17.0 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets)
(1.20.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.24.0-
>datasets) (4.13.2)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.11/dist-packages (from requests>=2.32.2-
>datasets) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in
```

```
/usr/local/lib/python3.11/dist-packages (from requests>=2.32.2-
>datasets) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.11/dist-packages (from requests>=2.32.2-
>datasets) (2.4.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.11/dist-packages (from requests>=2.32.2-
>datasets) (2025.1.31)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.11/dist-packages (from pandas->datasets)
(2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.11/dist-packages (from pandas->datasets)
(2025.2)
Requirement already satisfied: tzdata>=2022.7 in
/usr/local/lib/python3.11/dist-packages (from pandas->datasets)
(2025.2)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2-
>pandas->datasets) (1.17.0)
Downloading datasets-3.5.1-py3-none-any.whl (491 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 491.4/491.4 kB 27.9 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 116.3/116.3 kB 10.3 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 193.6/193.6 kB 18.6 MB/s eta
0:00:00
ultiprocess-0.70.16-py311-none-any.whl (143 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 143.5/143.5 kB 15.2 MB/s eta
0:00:00
anylinux_2_17_x86_64.manylinux2014_x86_64.whl (194 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 194.8/194.8 kB 18.4 MB/s eta
0:00:00
ultiprocess, datasets
  Attempting uninstall: fsspec
    Found existing installation: fsspec 2025.3.2
    Uninstalling fsspec-2025.3.2:
      Successfully uninstalled fsspec-2025.3.2
ERROR: pip's dependency resolver does not currently take into account
all the packages that are installed. This behaviour is the source of
the following dependency conflicts.
torch 2.6.0+cu124 requires nvidia-cublas-cu12==12.4.5.8;
platform_system == "Linux" and platform_machine == "x86_64", but you
have nvidia-cublas-cu12 12.5.3.2 which is incompatible.
torch 2.6.0+cu124 requires nvidia-cuda-cupti-cu12==12.4.127;
platform_system == "Linux" and platform_machine == "x86_64", but you
have nvidia-cuda-cupti-cu12 12.5.82 which is incompatible.
torch 2.6.0+cu124 requires nvidia-cuda-nvrtc-cu12==12.4.127;
platform_system == "Linux" and platform_machine == "x86_64", but you
```

```
have nvidia-cuda-nvrtc-cu12 12.5.82 which is incompatible.
torch 2.6.0+cu124 requires nvidia-cuda-runtime-cu12==12.4.127;
platform_system == "Linux" and platform_machine == "x86_64", but you
have nvidia-cuda-runtime-cu12 12.5.82 which is incompatible.
torch 2.6.0+cu124 requires nvidia-cudnn-cu12==9.1.0.70;
platform_system == "Linux" and platform_machine == "x86_64", but you
have nvidia-cudnn-cu12 9.3.0.75 which is incompatible.
torch 2.6.0+cu124 requires nvidia-cufft-cu12==11.2.1.3;
platform_system == "Linux" and platform_machine == "x86_64", but you
have nvidia-cufft-cu12 11.2.3.61 which is incompatible.
torch 2.6.0+cu124 requires nvidia-curand-cu12==10.3.5.147;
platform_system == "Linux" and platform_machine == "x86_64", but you
have nvidia-curand-cu12 10.3.6.82 which is incompatible.
torch 2.6.0+cu124 requires nvidia-cusolver-cu12==11.6.1.9;
platform_system == "Linux" and platform_machine == "x86_64", but you
have nvidia-cusolver-cu12 11.6.3.83 which is incompatible.
torch 2.6.0+cu124 requires nvidia-cusparse-cu12==12.3.1.170;
platform_system == "Linux" and platform_machine == "x86_64", but you
have nvidia-cusparse-cu12 12.5.1.3 which is incompatible.
torch 2.6.0+cu124 requires nvidia-nvjitlink-cu12==12.4.127;
platform_system == "Linux" and platform_machine == "x86_64", but you
have nvidia-nvjitlink-cu12 12.5.82 which is incompatible.
gcsfs 2025.3.2 requires fsspec==2025.3.2, but you have fsspec 2025.3.0
which is incompatible.
Successfully installed datasets-3.5.1 dill-0.3.8 fsspec-2025.3.0
multiprocess-0.70.16 xxhash-3.5.0

# parameters
MODEL_NAME   = "bert-base-uncased"
BATCH_SIZE   = 16
EPOCHS       = 6
LR           = 3e-5
WARMUP_RATIO = 0.1
THRESHOLD    = 0.4
LOG_STEP     = 100

ds = load_dataset("go_emotions", "simplified")  # 28 emotions
label_names = ds["train"].features["labels"].feature.names

NUM_LABELS = 28


tokenizer = BertTokenizerFast.from_pretrained(MODEL_NAME)
def tokenize_and_encode(examples):
    encodings = tokenizer(examples["text"], truncation=True)

    # shape = (batch_in_map, 28)
    multi_hot = np.zeros((len(examples["labels"]), NUM_LABELS),
dtype=np.int8)
    for i, label_list in enumerate(examples["labels"]):
```

```
        multi_hot[i, label_list] = 1

    encodings["labels"] = multi_hot.tolist()
    return encodings

ds = ds.map(tokenize_and_encode, batched=True, remove_columns=["text",
"id"])
collator = DataCollatorWithPadding(tokenizer, return_tensors="pt")

# DataLoader
train_loader = DataLoader(ds["train"], batch_size=BATCH_SIZE,
                          shuffle=True, collate_fn=collator)
val_loader = DataLoader(ds["validation"], batch_size=BATCH_SIZE,
                          shuffle=False, collate_fn=collator)
print(ds["train"].column_names)
print(len(ds["validation"]))
```

```
['labels', 'input_ids', 'token_type_ids', 'attention_mask']
5426
```

```
# define model
model = BertForSequenceClassification.from_pretrained(
    MODEL_NAME,
    num_labels=NUM_LABELS,
    problem_type="multi_label_classification",
).to(device)

model.init_weights()
```

```
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.
```

```
optimizer = AdamW(model.parameters(), lr=LR)

total_steps   = len(train_loader) * EPOCHS
warmup_steps  = int(total_steps * WARMUP_RATIO)
scheduler     = get_linear_schedule_with_warmup(
    optimizer, num_warmup_steps=warmup_steps,
num_training_steps=total_steps
)

'''
# label weight
label_array = np.array(ds["train"]["labels"])
label_freq = label_array.sum(axis=0)
pos_weight = torch.tensor(np.log((label_freq.max() + 1) / (label_freq
+ 1e-6)), device=device)
criterion = nn.BCEWithLogitsLoss(pos_weight=pos_weight)
```

```
'''

criterion = nn.BCEWithLogitsLoss()

from sklearn.metrics import accuracy_score, precision_score,
recall_score

#Evaluate
def evaluate(model, loader):
    model.eval()
    all_logits, all_labels = [], []
    with torch.no_grad():
        for batch in loader:
            labels =
batch["labels"].clone().detach().to(device).float()

            inputs = {k: v.to(device) for k, v in batch.items() if k !
= "labels"}
            logits = model(**inputs).logits.cpu()

            all_logits.append(logits)
            all_labels.append(labels)

    logits = torch.cat(all_logits)
    labels = torch.cat(all_labels)

    preds  = (torch.sigmoid(logits) > THRESHOLD).int().numpy()
    labels = labels.int().cpu().numpy()

    # accuracy, precision, recall
    acc = accuracy_score(labels, preds)
    precision = precision_score(labels, preds, average="micro",
zero_division=0)
    recall  = recall_score(labels, preds, average="micro",
zero_division=0)

    #f1
    f1 = 2 * precision * recall / (precision + recall + 1e-8)
    # stats
    print("average 1's in data", labels.sum(axis=1).mean())
    print("average 1's in prediction:", preds.sum(axis=1).mean())

    return acc, precision, recall, f1

# train
global_step = 0
for epoch in range(1, EPOCHS+1):
    model.train()
    epoch_loss = 0.0

    prog_bar = tqdm(train_loader, desc=f"Epoch {epoch}", leave=False)
```

```
    for step, batch in enumerate(prog_bar, 1):
        batch_size = len(batch["labels"])
        labels = batch["labels"].clone().detach().to(device).float()

        inputs = {k: v.to(device) for k, v in batch.items() if k != 
"labels"}

        # zero grad
        optimizer.zero_grad()

        # forward
        outputs = model(**inputs)
        loss = criterion(outputs.logits, labels)
        loss.backward()

        optimizer.step()
        scheduler.step()

        epoch_loss += loss.item()
        global_step += 1

        if global_step % LOG_STEP == 0:
            prog_bar.set_postfix(loss=f"{loss.item():.4f}")

    avg_loss = epoch_loss / len(train_loader)
    acc, prec, recall, f1 = evaluate(model, val_loader)

    print(
        f"Epoch {epoch} |"
        f"loss {avg_loss:.4f} | "
        f"accuracy {acc:.4f} | "
        f"Precision {prec:.4f} | "
        f"Recall {recall:.4f} | "
        f"F1 {f1:.4f}")
```

{"model_id":"1422720a770e467d94ea67532e5799db","version_major":2,"vers
ion_minor":0}

```
average 1's in data 1.1758201253225211
average 1's in prediction: 0.6929598230740878
Epoch 1 |loss 0.1797 | accuracy 0.3972 | Precision 0.6931 | Recall 
0.4085 | F1 0.5140
```

{"model_id":"eb1138632f6a4a7f801f5737c23e053f","version_major":2,"vers
ion_minor":0}

```
average 1's in data 1.1758201253225211
average 1's in prediction: 0.9235164025064504
Epoch 2 |loss 0.0880 | accuracy 0.4838 | Precision 0.6711 | Recall 
0.5271 | F1 0.5905
```

{"model_id":"c6c26554a6bc49bbb8ace81068d7fda9","version_major":2,"version_minor":0}

```
average 1's in data 1.1758201253225211
average 1's in prediction: 1.0081091043125692
Epoch 3 |loss 0.0710 | accuracy 0.4799 | Precision 0.6483 | Recall 0.5558 | F1 0.5985
```

{"model_id":"3cf803def44043dc9a4d4ba57efe2a61","version_major":2,"version_minor":0}

```
average 1's in data 1.1758201253225211
average 1's in prediction: 1.1098415038702543
Epoch 4 |loss 0.0558 | accuracy 0.4641 | Precision 0.6025 | Recall 0.5687 | F1 0.5851
```

{"model_id":"de7a1daf3e7141d78b9a51a308c3dce8","version_major":2,"version_minor":0}

```
average 1's in data 1.1758201253225211
average 1's in prediction: 1.1126059712495393
Epoch 5 |loss 0.0437 | accuracy 0.4624 | Precision 0.5940 | Recall 0.5621 | F1 0.5776
```

{"model_id":"1844822ee34843579b85307f97fdb5a5","version_major":2,"version_minor":0}

```
average 1's in data 1.1758201253225211
average 1's in prediction: 1.1262440103206781
Epoch 6 |loss 0.0359 | accuracy 0.4582 | Precision 0.5896 | Recall 0.5647 | F1 0.5769
```

```
'''
def sweep_thresholds(model, loader, start=0.3, stop=0.6, step=0.05, limit=500):
    print("\n--- Threshold Sweep (first", limit, "samples) ---")
    model.eval()
    all_logits, all_labels = [], []
    count = 0
    with torch.no_grad():
        for batch in loader:
            labels = batch["labels"].clone().detach().to(device).float()
            inputs = {k: v.to(device) for k, v in batch.items() if k != "labels"}
            logits = model(**inputs).logits.detach().cpu()
            all_logits.append(logits)
            all_labels.append(labels.cpu())
            count += len(labels)
            if count >= limit:
                break
```

```
    logits = torch.cat(all_logits)[:limit]
    labels = torch.cat(all_labels)[:limit]

    for t in np.arange(start, stop + step, step):
        preds = (torch.sigmoid(logits) > t).int().numpy()
        targets = labels.int().numpy()
        acc = accuracy_score(targets, preds)
        precision = precision_score(targets, preds, average="micro",
zero_division=0)
        recall = recall_score(targets, preds, average="micro",
zero_division=0)
        f1 = 2 * precision * recall / (precision + recall + 1e-8)
        print("average 1's in data", labels.sum(axis=1).mean())
        print("average 1's in prediction:", preds.sum(axis=1).mean())
        print(f"Threshold = {t:.3f} | Precision = {precision:.4f} |
Recall = {recall:.4f} | F1 = {f1:.4f}")

sweep_thresholds(model, val_loader, start=0.3, stop=0.7, step=0.05,
limit=500)
'''

os.makedirs("saved_models", exist_ok=True)
torch.save(model.state_dict(), "saved_models/bert_goemotion_f1_58.pt")
```