```
!pip install --upgrade transformers datasets accelerate
from google.colab import drive
drive.mount('/content/drive')
from sklearn.metrics import accuracy_score, precision_recall_fscore_support
import transformers
import pandas as pd
from datasets import Dataset
from transformers import BertTokenizer, BertForSequenceClassification, Trainer,
TrainingArguments
from sklearn.model_selection import train_test_split
import torch
```

Show hidden output

```
def compute_metrics(pred):
    labels = pred.label_ids
    preds = pred.predictions.argmax(-1)
    acc = accuracy_score(labels, preds)
    precision, recall, f1, _ = precision_recall_fscore_support(labels, preds, average='binary')
    return {
        'accuracy': acc,
        'precision': precision,
        'recall': recall,
        'f1': f1
    }
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
# Only use the input text and hate label
df = pd.read_csv('/content/drive/MyDrive/train_with_topics.csv')
df = df[['text', 'topic_label', 'hate_label']].dropna()
df['hate_label'] = df['hate_label'].astype(int)

# Split into train/test
from sklearn.model_selection import train_test_split
train_df, val_df = train_test_split(df, test_size=0.1, random_state=42)

# Convert to Hugging Face dataset
train_dataset = Dataset.from_pandas(train_df)
val_dataset = Dataset.from_pandas(val_df)
full_dataset = Dataset.from_pandas(df)
```

```
tokenizer = BertTokenizer.from_pretrained("bert-base-uncased")

def tokenize(example):
    return tokenizer(example['text'], truncation=True, padding="max_length", max_length=128)

train_dataset = train_dataset.map(tokenize, batched=True)
val_dataset = val_dataset.map(tokenize, batched=True)
full_dataset = full_dataset.map(tokenize, batched=True)

train_dataset = train_dataset.rename_column("hate_label", "labels")
val_dataset = val_dataset.rename_column("hate_label", "labels")
full_dataset = full_dataset.rename_column("hate_label", "labels")

train_dataset.set_format("torch")
val_dataset.set_format("torch")
full_dataset.set_format("torch")

```

```
Map: 100%                                    13927/13927 [00:08<00:00, 1621.94 examples/s]

Map: 100%                                    1548/1548 [00:00<00:00, 1679.21 examples/s]

Map: 100%                                    15475/15475 [00:09<00:00, 1727.65 examples/s]
```

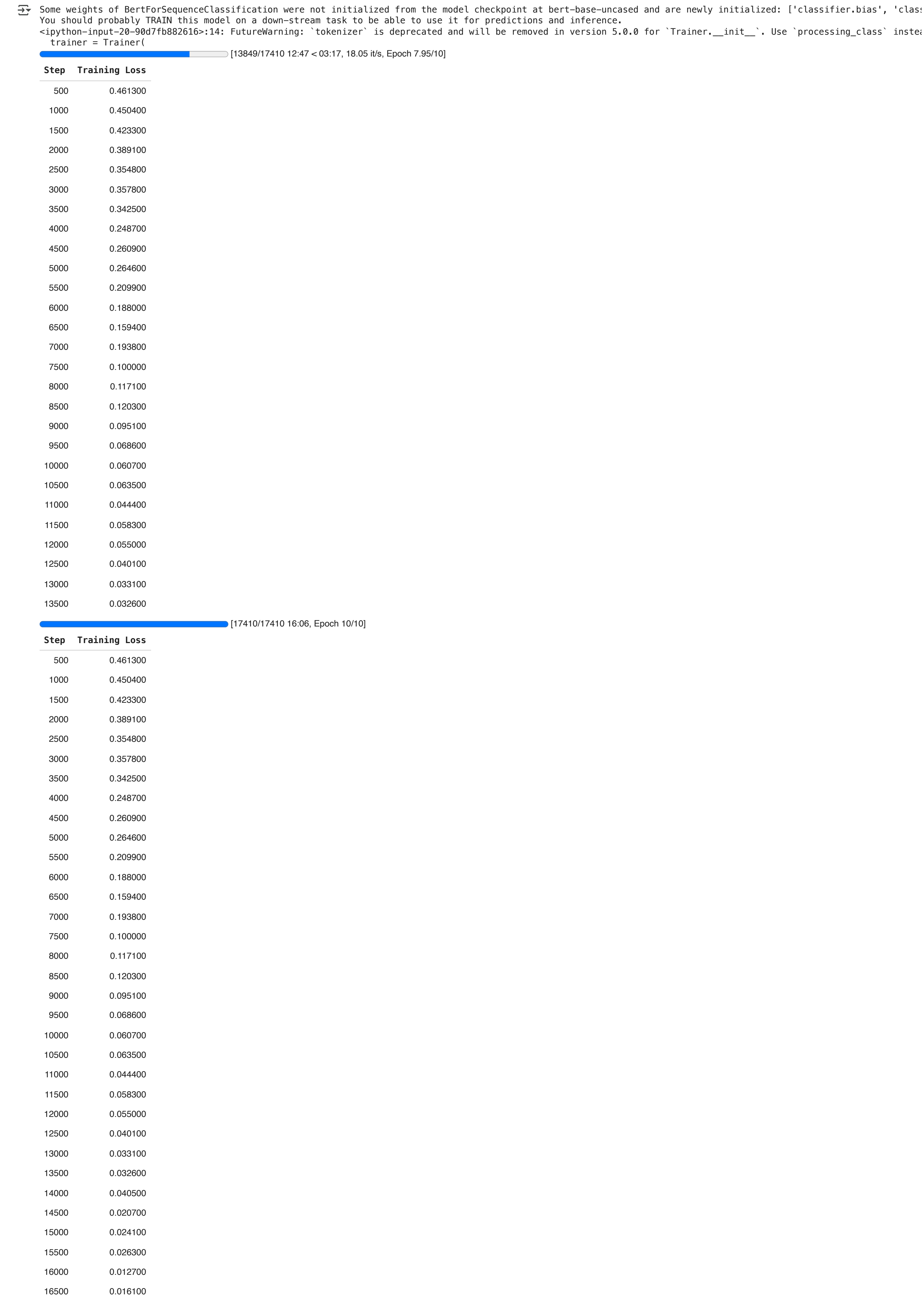
```
# Load model
model = BertForSequenceClassification.from_pretrained("bert-base-uncased", num_labels=2)

# Define training arguments
training_args = TrainingArguments(
    output_dir="/content/drive/MyDrive/hate_model_basic_full_dataset",
    save_strategy="epoch",
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    num_train_epochs=10,
    logging_dir="/content/logs"
)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=val_dataset,
    tokenizer=tokenizer,
    compute_metrics=compute_metrics
)

# Train and evaluate
trainer.train()
trainer.evaluate()

```



```
Some weights of BertForSequenceClassification were not initialized from the model checkpoint at bert-base-uncased and are newly initialized: ['classifier.bias', 'class
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
<ipython-input-20-90d7fb882616>:14: FutureWarning: `tokenizer` is deprecated and will be removed in version 5.0.0 for `Trainer.__init__`. Use `processing_class` instea
  trainer = Trainer(
```

[13849/17410 12:47 < 03:17, 18.05 it/s, Epoch 7.95/10]

Step	Training Loss
500	0.461300
1000	0.450400
1500	0.423300
2000	0.389100
2500	0.354800
3000	0.357800
3500	0.342500
4000	0.248700
4500	0.260900
5000	0.264600
5500	0.209900
6000	0.188000
6500	0.159400
7000	0.193800
7500	0.100000
8000	0.117100
8500	0.120300
9000	0.095100
9500	0.068600
10000	0.060700
10500	0.063500
11000	0.044400
11500	0.058300
12000	0.055000
12500	0.040100
13000	0.033100
13500	0.032600

[17410/17410 16:06, Epoch 10/10]

Step	Training Loss
500	0.461300
1000	0.450400
1500	0.423300
2000	0.389100
2500	0.354800
3000	0.357800
3500	0.342500
4000	0.248700
4500	0.260900
5000	0.264600
5500	0.209900
6000	0.188000
6500	0.159400
7000	0.193800
7500	0.100000
8000	0.117100
8500	0.120300
9000	0.095100
9500	0.068600
10000	0.060700
10500	0.063500
11000	0.044400
11500	0.058300
12000	0.055000
12500	0.040100
13000	0.033100
13500	0.032600
14000	0.040500
14500	0.020700
15000	0.024100
15500	0.026300
16000	0.012700
16500	0.016100

```
17000        0.020200
```

```
[194/194 00:03]
{'eval_loss': 1.321150779724121,
 'eval_accuracy': 0.8275193798449613,
 'eval_precision': 0.625,
 'eval_recall': 0.5539358600583091,
 'eval_f1': 0.5873261205564142,
 'eval_runtime': 3.1807,
 'eval_samples_per_second': 486.685,
 'eval_steps_per_second': 60.993,
 'epoch': 10.0}
```

```python
# Running model in-domain on Twitter dataset
# Load tokenizer
tokenizer = BertTokenizer.from_pretrained("bert-base-uncased")

# Load test CSV
df = pd.read_csv("/content/drive/MyDrive/train.csv")

# Batch size for prediction
batch_size = 32

# Get the device
device = "cuda" if torch.cuda.is_available() else "cpu"
model.to(device)  # Ensure model is on the correct device

# Store predictions
all_preds = []
all_probs = []

# Iterate over the data in batches
for i in range(0, len(df), batch_size):
    batch = df[i : i + batch_size]

    test_encodings = tokenizer(
        batch["text"].tolist(),
        truncation=True,
        padding=True,
        max_length=128,
        return_tensors="pt",
    ).to(device)

    model.eval()
    with torch.no_grad():
        outputs = model(**test_encodings)
        probs = torch.softmax(outputs.logits, dim=1)
        preds = torch.argmax(probs, dim=1)

    all_preds.extend(preds.cpu().numpy())
    all_probs.extend(probs.cpu().numpy().tolist())

# Add predictions to DataFrame
df["predicted_label"] = all_preds
df["predicted_prob"] = all_probs

# Calculate accuracy
correct = (df["predicted_label"] == df["hate_label"]).sum()
accuracy = correct / len(df)

# Print results
print(df[["text", "hate_label", "predicted_label"]].head())
print(f"Accuracy: {accuracy:.4f}")

```

```
                                                text  hate_label  \
0  The trans women reading this tweet right now i...           0
1  9) uhhhh i like being lgbt a lot. i feel proud...           0
2  @terryelaineh1 @UKLabour Why do 3.8 million #5...           0
3  I said it yesterday, I knew this is about to g...           0
4  White Small Little Invisible Clits Are A Disgr...           1

   predicted_label
0                0
1                0
2                0
3                0
4                1
Accuracy: 0.9899
```

```python
 # Running model out of domain on Reddit dataset

 # Load tokenizer
 tokenizer = BertTokenizer.from_pretrained("bert-base-uncased")

 # Load test CSV
 df = pd.read_csv("/content/drive/MyDrive/test_reddit.csv")

 # Get the device
 device = "cuda" if torch.cuda.is_available() else "cpu"
 model.to(device)  # Ensure model is on the correct device

 # Batch size for prediction
 batch_size = 32

 # Store predictions
 all_preds = []
 all_probs = []

 # Iterate over the data in batches
 for i in range(0, len(df), batch_size):
     batch = df[i : i + batch_size]

     test_encodings = tokenizer(
         batch["text"].tolist(),
         truncation=True,
         padding=True,
         max_length=128,
         return_tensors="pt",
     ).to(device)
```

```
    model.eval()
    with torch.no_grad():
        outputs = model(**test_encodings)
        probs = torch.softmax(outputs.logits, dim=1)
        preds = torch.argmax(probs, dim=1)

    all_preds.extend(preds.cpu().numpy())
    all_probs.extend(probs.cpu().numpy().tolist())

# Add predictions to DataFrame
df["predicted_label"] = all_preds
df["predicted_prob"] = all_probs

# Calculate accuracy
correct = (df["predicted_label"] == df["hate_label"]).sum()
accuracy = correct / len(df)

# Print results
print(df[["text", "hate_label", "predicted_label"]].head())
print(f"Accuracy: {accuracy:.4f}")
```

```
                                                text  hate_label  \
0  For starters bend over the one in pink and kic...           1
1  Sounds like the kinda wholsesome life I'd die ...           0
2  Who the fuck is this insignificant simple mind...           0
3         Fuck off you insufferable retarded faggot.           1
4  Worthless whore, these tits with look nice wit...           0

   predicted_label
0                0
1                0
2                1
3                1
4                1
Accuracy: 0.7957
```

```
# Running model out of domain on Youtube dataset

# Load tokenizer
tokenizer = BertTokenizer.from_pretrained("bert-base-uncased")

# Load test CSV
df = pd.read_csv("/content/drive/MyDrive/test_youtube.csv")

# Batch size for prediction
batch_size = 32

# Store predictions
all_preds = []
all_probs = []

# Iterate over the data in batches
for i in range(0, len(df), batch_size):
    batch = df[i : i + batch_size]

    test_encodings = tokenizer(
        batch["text"].tolist(),
        truncation=True,
        padding=True,
        max_length=128,
        return_tensors="pt",
    ).to(device)

    model.eval()
    with torch.no_grad():
        outputs = model(**test_encodings)
        probs = torch.softmax(outputs.logits, dim=1)
        preds = torch.argmax(probs, dim=1)

    all_preds.extend(preds.cpu().numpy())
    all_probs.extend(probs.cpu().numpy().tolist())

# Add predictions to DataFrame
df["predicted_label"] = all_preds
df["predicted_prob"] = all_probs

# Calculate accuracy
correct = (df["predicted_label"] == df["hate_label"]).sum()
accuracy = correct / len(df)

# Print results
print(df[["text", "hate_label", "predicted_label"]].head())
print(f"Accuracy: {accuracy:.4f}")
```

```
                                                text  hate_label  \
0  Yes indeed. She sort of reminds me of the elde...           0
1  Question: These 4 broads who criticize America...           0
2  It is about time for all illegals to go back t...           0
3  OMG! The EGO's of these young, young, inexperi...           0
4  Joshua Lelo so you have seen all actors from e...           0

   predicted_label
0                0
1                0
2                0
3                0
4                0
Accuracy: 0.7754
```