Starter Notebook

Install and import required libraries

```
!pip install transformers datasets evaluate accelerate peft trl bitsandbytes
!pip install nvidia-ml-py3
!pip install scikit-learn
```

```
Requirement already satisfied: safetensors>=0.4.3 in /home/ab12660/.local/li
Requirement already satisfied: tqdm>=4.27 in /home/ab12660/.local/lib/pythor
Requirement already satisfied: pyarrow>=15.0.0 in /home/ab12660/.local/lib/p
Requirement already satisfied: dill<0.3.9,>=0.3.0 in /home/ab12660/.local/li
Requirement already satisfied: pandas in /home/ab12660/.local/lib/python3.9/
Requirement already satisfied: xxhash in /home/ab12660/.local/lib/python3.9/
Requirement already satisfied: multiprocess<0.70.17 in /home/ab12660/.local/
Requirement already satisfied: fsspec<=2024.12.0,>=2023.1.0 in /home/ab12660
Requirement already satisfied: aiohttp in /home/ab12660/.local/lib/python3.9
Requirement already satisfied: psutil in /share/apps/pyenv/py3.9/lib/python3
Requirement already satisfied: torch>=2.0.0 in /home/ab12660/.local/lib/pyth
Requirement already satisfied: rich in /home/ab12660/.local/lib/python3.9/si
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /home/ab12660/.loc
Requirement already satisfied: aiosignal>=1.1.2 in /home/ab12660/.local/lib/
Requirement already satisfied: async-timeout<6.0,>=4.0 in /home/ab12660/.loc
Requirement already satisfied: attrs>=17.3.0 in /share/apps/pyenv/py3.9/lib/
Requirement already satisfied: frozenlist>=1.1.1 in /home/ab12660/.local/lib
Requirement already satisfied: multidict<7.0,>=4.5 in /home/ab12660/.local/l
Requirement already satisfied: propcache>=0.2.0 in /home/ab12660/.local/lib/
Requirement already satisfied: yarl<2.0,>=1.17.0 in /home/ab12660/.local/lib
Requirement already satisfied: typing-extensions>=3.7.4.3 in /home/ab12660/.
Requirement already satisfied: charset-normalizer<4,>=2 in /share/apps/pyenv
Requirement already satisfied: idna<4,>=2.5 in /share/apps/pyenv/py3.9/lib/p
Requirement already satisfied: urllib3<3,>=1.21.1 in /share/apps/pyenv/py3.9
Requirement already satisfied: certifi>=2017.4.17 in /share/apps/pyenv/py3.9
Requirement already satisfied: networkx in /home/ab12660/.local/lib/python3.
Requirement already satisfied: jinja2 in /share/apps/pyenv/py3.9/lib/python3
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.4.127 in /home/ab1
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.4.127 in /home/a
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.4.127 in /home/ab1
Requirement already satisfied: nvidia-cudnn-cu12==9.1.0.70 in /home/ab12660/
Requirement already satisfied: nvidia-cublas-cu12==12.4.5.8 in /home/ab12660
Requirement already satisfied: nvidia-cufft-cu12==11.2.1.3 in /home/ab12660/
Requirement already satisfied: nvidia-curand-cu12==10.3.5.147 in /home/ab126
Requirement already satisfied: nvidia-cusolver-cu12==11.6.1.9 in /home/ab126
Requirement already satisfied: nvidia-cusparse-cu12==12.3.1.170 in /home/ab1
Requirement already satisfied: nvidia-cusparselt-cu12==0.6.2 in /home/ab1266
Requirement already satisfied: nvidia-nccl-cu12==2.21.5 in /home/ab12660/.lo
Requirement already satisfied: nvidia-nvtx-cu12==12.4.127 in /home/ab12660/.
```

```
Requirement already satisfied: nvidia-nvjitlink-cu12==12.4.127 in /home/ab12
Requirement already satisfied: triton==3.2.0 in /home/ab12660/.local/lib/pyt
Requirement already satisfied: sympy==1.13.1 in /home/ab12660/.local/lib/pyt
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /home/ab12660/.local/li
Requirement already satisfied: python-dateutil>=2.8.2 in /share/apps/pyenv/p
Requirement already satisfied: pytz>=2020.1 in /home/ab12660/.local/lib/pyth
Requirement already satisfied: tzdata>=2022.7 in /home/ab12660/.local/lib/py
Requirement already satisfied: markdown-it-py>=2.2.0 in /home/ab12660/.local
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /share/apps/pyenv/
Requirement already satisfied: mdurl~=0.1 in /home/ab12660/.local/lib/python
Requirement already satisfied: six>=1.5 in /share/apps/pyenv/py3.9/lib/pytho
Requirement already satisfied: MarkupSafe>=2.0 in /share/apps/pyenv/py3.9/li
Defaulting to user installation because normal site-packages is not writeabl
Requirement already satisfied: nvidia-ml-py3 in /home/ab12660/.local/lib/pyt
Defaulting to user installation because normal site-packages is not writeabl
Requirement already satisfied: scikit-learn in /home/ab12660/.local/lib/pyth
Requirement already satisfied: numpy>=1.19.5 in /home/ab12660/.local/lib/pyt
Requirement already satisfied: scipy>=1.6.0 in /home/ab12660/.local/lib/pyth
Requirement already satisfied: joblib>=1.2.0 in /home/ab12660/.local/lib/pyt
Requirement already satisfied: threadpoolctl>=3.1.0 in /home/ab12660/.local/
```

```
!pip install os
!pip install pandas
!pip install torch
!pip install transformers
!pip install peft
!pip install datasets
!pip install pickle
!pip install --user pandas
```

```
Defaulting to user installation because normal site-packages is not writeabl
ERROR: Could not find a version that satisfies the requirement os (from vers
ERROR: No matching distribution found for os
Defaulting to user installation because normal site-packages is not writeabl
Requirement already satisfied: pandas in /home/ab12660/.local/lib/python3.9/
Requirement already satisfied: numpy>=1.22.4 in /home/ab12660/.local/lib/pyt
Requirement already satisfied: python-dateutil>=2.8.2 in /share/apps/pyenv/p
Requirement already satisfied: pytz>=2020.1 in /home/ab12660/.local/lib/pyth
Requirement already satisfied: tzdata>=2022.7 in /home/ab12660/.local/lib/py
Requirement already satisfied: six>=1.5 in /share/apps/pyenv/py3.9/lib/pytho
Defaulting to user installation because normal site-packages is not writeabl
Requirement already satisfied: torch in /home/ab12660/.local/lib/python3.9/s
Requirement already satisfied: filelock in /home/ab12660/.local/lib/python3.
Requirement already satisfied: typing-extensions>=4.10.0 in /home/ab12660/.l
Requirement already satisfied: networkx in /home/ab12660/.local/lib/python3.
Requirement already satisfied: jinja2 in /share/apps/pyenv/py3.9/lib/python3
Requirement already satisfied: fsspec in /home/ab12660/.local/lib/python3.9/
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.4.127 in /home/ab1
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.4.127 in /home/a
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.4.127 in /home/ab1
Requirement already satisfied: nvidia-cudnn-cu12==9.1.0.70 in /home/ab12660/
```

```
Requirement already satisfied: nvidia-cublas-cu12==12.4.5.8 in /home/ab12660
Requirement already satisfied: nvidia-cufft-cu12==11.2.1.3 in /home/ab12660/
Requirement already satisfied: nvidia-curand-cu12==10.3.5.147 in /home/ab126
Requirement already satisfied: nvidia-cusolver-cu12==11.6.1.9 in /home/ab126
Requirement already satisfied: nvidia-cusparse-cu12==12.3.1.170 in /home/ab1
Requirement already satisfied: nvidia-cusparselt-cu12==0.6.2 in /home/ab1266
Requirement already satisfied: nvidia-nccl-cu12==2.21.5 in /home/ab12660/.lo
Requirement already satisfied: nvidia-nvtx-cu12==12.4.127 in /home/ab12660/.
Requirement already satisfied: nvidia-nvjitlink-cu12==12.4.127 in /home/ab12
Requirement already satisfied: triton==3.2.0 in /home/ab12660/.local/lib/pyt
Requirement already satisfied: sympy==1.13.1 in /home/ab12660/.local/lib/pyt
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /home/ab12660/.local/li
Requirement already satisfied: MarkupSafe>=2.0 in /share/apps/pyenv/py3.9/li
Defaulting to user installation because normal site-packages is not writeabl
Requirement already satisfied: transformers in /home/ab12660/.local/lib/pyth
Requirement already satisfied: filelock in /home/ab12660/.local/lib/python3.
Requirement already satisfied: huggingface-hub<1.0,>=0.30.0 in /home/ab12660
Requirement already satisfied: numpy>=1.17 in /home/ab12660/.local/lib/pytho
Requirement already satisfied: packaging>=20.0 in /share/apps/pyenv/py3.9/li
Requirement already satisfied: pyyaml>=5.1 in /share/apps/pyenv/py3.9/lib/py
Requirement already satisfied: regex!=2019.12.17 in /home/ab12660/.local/lib
Requirement already satisfied: requests in /home/ab12660/.local/lib/python3.
Requirement already satisfied: tokenizers<0.22,>=0.21 in /home/ab12660/.loca
Requirement already satisfied: safetensors>=0.4.3 in /home/ab12660/.local/li
Requirement already satisfied: tqdm>=4.27 in /home/ab12660/.local/lib/python
Requirement already satisfied: fsspec>=2023.5.0 in /home/ab12660/.local/lib/
Requirement already satisfied: typing-extensions>=3.7.4.3 in /home/ab12660/.
Requirement already satisfied: charset-normalizer<4,>=2 in /share/apps/pyenv
Requirement already satisfied: idna<4,>=2.5 in /share/apps/pyenv/py3.9/lib/p
Requirement already satisfied: urllib3<3,>=1.21.1 in /share/apps/pyenv/py3.9
Requirement already satisfied: certifi>=2017.4.17 in /share/apps/pyenv/py3.9
Defaulting to user installation because normal site-packages is not writeabl
Requirement already satisfied: peft in /home/ab12660/.local/lib/python3.9/si
Requirement already satisfied: numpy>=1.17 in /home/ab12660/.local/lib/pytho
Requirement already satisfied: packaging>=20.0 in /share/apps/pyenv/py3.9/li
Requirement already satisfied: psutil in /share/apps/pyenv/py3.9/lib/python3
Requirement already satisfied: pyyaml in /share/apps/pyenv/py3.9/lib/python3
Requirement already satisfied: torch>=1.13.0 in /home/ab12660/.local/lib/pyt
```

```
!pip install pandas
import os
import pandas as pd
import torch
from transformers import RobertaModel, RobertaTokenizer, TrainingArguments, Train
from peft import LoraConfig, get_peft_model, PeftModel
from datasets import load_dataset, Dataset, ClassLabel
import pickle
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: pandas in /home/ab12660/.local/lib/python3.9/s
Requirement already satisfied: numpy>=1.22.4 in /home/ab12660/.local/lib/pytho
Requirement already satisfied: python-dateutil>=2.8.2 in /share/apps/pyenv/py3
Requirement already satisfied: pytz>=2020.1 in /home/ab12660/.local/lib/python
Requirement already satisfied: tzdata>=2022.7 in /home/ab12660/.local/lib/pyth
Requirement already satisfied: six>=1.5 in /share/apps/pyenv/py3.9/lib/python3
/home/ab12660/.local/lib/python3.9/site-packages/tqdm/auto.py:21: TqdmWarning:
  from .autonotebook import tqdm as notebook_tqdm
```

Load Tokenizer and Preprocess Data

Start coding or generate with AI.

```
base_model = 'roberta-base'

dataset = load_dataset('ag_news', split='train')
tokenizer = RobertaTokenizer.from_pretrained(base_model)

def preprocess(examples):
    tokenized = tokenizer(examples['text'], truncation=True, padding=True)
    return tokenized

tokenized_dataset = dataset.map(preprocess, batched=True,  remove_columns=["text"
tokenized_dataset = tokenized_dataset.rename_column("label", "labels")
```

```
# Extract the number of classess and their names
num_labels = dataset.features['label'].num_classes
class_names = dataset.features["label"].names
print(f"number of labels: {num_labels}")
print(f"the labels: {class_names}")

# Create an id2label mapping
# We will need this for our classifier.
id2label = {i: label for i, label in enumerate(class_names)}

data_collator = DataCollatorWithPadding(tokenizer=tokenizer, return_tensors="pt")
```

```
number of labels: 4
the labels: ['World', 'Sports', 'Business', 'Sci/Tech']
```

Load Pre-trained Model

Set up config for pretrained model and download it from hugging face

```
model = RobertaForSequenceClassification.from_pretrained(
    base_model,
    id2label=id2label)
model
```

```
Some weights of RobertaForSequenceClassification were not initialized from the
You should probably TRAIN this model on a down-stream task to be able to use i
RobertaForSequenceClassification(
  (roberta): RobertaModel(
    (embeddings): RobertaEmbeddings(
      (word_embeddings): Embedding(50265, 768, padding_idx=1)
      (position_embeddings): Embedding(514, 768, padding_idx=1)
      (token_type_embeddings): Embedding(1, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (encoder): RobertaEncoder(
      (layer): ModuleList(
        (0-11): 12 x RobertaLayer(
          (attention): RobertaAttention(
            (self): RobertaSdpaSelfAttention(
              (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): RobertaSelfOutput(
              (dense): Linear(in_features=768, out_features=768, bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-05,
elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): RobertaIntermediate(
            (dense): Linear(in_features=768, out_features=3072, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): RobertaOutput(
            (dense): Linear(in_features=3072, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-05,
elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
      )
    )
  )
  (classifier): RobertaClassificationHead(
    (dense): Linear(in_features=768, out_features=768, bias=True)
    (dropout): Dropout(p=0.1, inplace=False)
    (out_proj): Linear(in_features=768, out_features=4, bias=True)
  )
)
```

Anything from here on can be modified

```
# Split the original training set
split_datasets = tokenized_dataset.train_test_split(test_size=640, seed=42)
train_dataset = split_datasets['train']
eval_dataset = split_datasets['test']
```

Setup LoRA Config

Setup PEFT config and get peft model for finetuning

```
# PEFT Config
# Replace old LoRA config with this
peft_config = LoraConfig(
    r=10,                           # Try r=10 for stronger adaptation
    lora_alpha=40,                # Stronger low-rank scaling
    lora_dropout=0.05,
    bias='none',
    target_modules=["query", "value"],  # Include both query and value projection
    task_type="SEQ_CLS"
)
```

```
peft_model = get_peft_model(model, peft_config)
peft_model
```

```
PeftModelForSequenceClassification(
  (base_model): LoraModel(
    (model): RobertaForSequenceClassification(
      (roberta): RobertaModel(
        (embeddings): RobertaEmbeddings(
          (word_embeddings): Embedding(50265, 768, padding_idx=1)
          (position_embeddings): Embedding(514, 768, padding_idx=1)
          (token_type_embeddings): Embedding(1, 768)
          (LayerNorm): LayerNorm((768,), eps=1e-05,
    elementwise_affine=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
        (encoder): RobertaEncoder(
          (layer): ModuleList(
            (0-11): 12 x RobertaLayer(
              (attention): RobertaAttention(
                (self): RobertaSdpaSelfAttention(
                  (query): lora.Linear(
```

```
                    (base_layer): Linear(in_features=768, out_features=768,
bias=True)
                    (lora_dropout): ModuleDict(
                      (default): Dropout(p=0.05, inplace=False)
                    )
                    (lora_A): ModuleDict(
                      (default): Linear(in_features=768, out_features=10,
bias=False)
                    )
                    (lora_B): ModuleDict(
                      (default): Linear(in_features=10, out_features=768,
bias=False)
                    )
                    (lora_embedding_A): ParameterDict()
                    (lora_embedding_B): ParameterDict()
                    (lora_magnitude_vector): ModuleDict()
                  )
                  (key): Linear(in_features=768, out_features=768,
bias=True)
                  (value): lora.Linear(
                    (base_layer): Linear(in_features=768, out_features=768,
bias=True)
                    (lora_dropout): ModuleDict(
                      (default): Dropout(p=0.05, inplace=False)
                    )
                    (lora_A): ModuleDict(
                      (default): Linear(in_features=768, out_features=10,
bias=False)
                    )
                    (lora_B): ModuleDict(
                      (default): Linear(in_features=10, out_features=768,
bias=False)
                    )
                    (lora_embedding_A): ParameterDict()
                    (lora_embedding_B): ParameterDict()
                    (lora_magnitude_vector): ModuleDict()
                  )
                  (dropout): Dropout(p=0.1, inplace=False)
                )
                (output): RobertaSelfOutput(
                  (dense): Linear(in_features=768, out_features=768,
```

```
print("Trainable parameters:")
for name, param in peft_model.named_parameters():
    if param.requires_grad:
        print(name)
```

```
Trainable parameters:
base_model.model.roberta.encoder.layer.0.attention.self.query.lora_A.default.w
base_model.model.roberta.encoder.layer.0.attention.self.query.lora_B.default.w
base_model.model.roberta.encoder.layer.0.attention.self.value.lora_A.default.w
base_model.model.roberta.encoder.layer.0.attention.self.value.lora_B.default.w
base_model.model.roberta.encoder.layer.1.attention.self.query.lora_A.default.w
```

base_model.model.roberta.encoder.layer.1.attention.self.query.lora_B.default.w
base_model.model.roberta.encoder.layer.1.attention.self.value.lora_A.default.w
base_model.model.roberta.encoder.layer.1.attention.self.value.lora_B.default.w
base_model.model.roberta.encoder.layer.2.attention.self.query.lora_A.default.w
base_model.model.roberta.encoder.layer.2.attention.self.query.lora_B.default.w
base_model.model.roberta.encoder.layer.2.attention.self.value.lora_A.default.w
base_model.model.roberta.encoder.layer.2.attention.self.value.lora_B.default.w
base_model.model.roberta.encoder.layer.3.attention.self.query.lora_A.default.w
base_model.model.roberta.encoder.layer.3.attention.self.query.lora_B.default.w
base_model.model.roberta.encoder.layer.3.attention.self.value.lora_A.default.w
base_model.model.roberta.encoder.layer.3.attention.self.value.lora_B.default.w
base_model.model.roberta.encoder.layer.4.attention.self.query.lora_A.default.w
base_model.model.roberta.encoder.layer.4.attention.self.query.lora_B.default.w
base_model.model.roberta.encoder.layer.4.attention.self.value.lora_A.default.w
base_model.model.roberta.encoder.layer.4.attention.self.value.lora_B.default.w
base_model.model.roberta.encoder.layer.5.attention.self.query.lora_A.default.w
base_model.model.roberta.encoder.layer.5.attention.self.query.lora_B.default.w
base_model.model.roberta.encoder.layer.5.attention.self.value.lora_A.default.w
base_model.model.roberta.encoder.layer.5.attention.self.value.lora_B.default.w
base_model.model.roberta.encoder.layer.6.attention.self.query.lora_A.default.w
base_model.model.roberta.encoder.layer.6.attention.self.query.lora_B.default.w
base_model.model.roberta.encoder.layer.6.attention.self.value.lora_A.default.w
base_model.model.roberta.encoder.layer.6.attention.self.value.lora_B.default.w
base_model.model.roberta.encoder.layer.7.attention.self.query.lora_A.default.w
base_model.model.roberta.encoder.layer.7.attention.self.query.lora_B.default.w
base_model.model.roberta.encoder.layer.7.attention.self.value.lora_A.default.w
base_model.model.roberta.encoder.layer.7.attention.self.value.lora_B.default.w
base_model.model.roberta.encoder.layer.8.attention.self.query.lora_A.default.w
base_model.model.roberta.encoder.layer.8.attention.self.query.lora_B.default.w
base_model.model.roberta.encoder.layer.8.attention.self.value.lora_A.default.w
base_model.model.roberta.encoder.layer.8.attention.self.value.lora_B.default.w
base_model.model.roberta.encoder.layer.9.attention.self.query.lora_A.default.w
base_model.model.roberta.encoder.layer.9.attention.self.query.lora_B.default.w
base_model.model.roberta.encoder.layer.9.attention.self.value.lora_A.default.w
base_model.model.roberta.encoder.layer.9.attention.self.value.lora_B.default.w
base_model.model.roberta.encoder.layer.10.attention.self.query.lora_A.default.
base_model.model.roberta.encoder.layer.10.attention.self.query.lora_B.default.
base_model.model.roberta.encoder.layer.10.attention.self.value.lora_A.default.
base_model.model.roberta.encoder.layer.10.attention.self.value.lora_B.default.
base_model.model.roberta.encoder.layer.11.attention.self.query.lora_A.default.
base_model.model.roberta.encoder.layer.11.attention.self.query.lora_B.default.
base_model.model.roberta.encoder.layer.11.attention.self.value.lora_A.default.
base_model.model.roberta.encoder.layer.11.attention.self.value.lora_B.default.
base_model.model.classifier.modules_to_save.default.dense.weight
base_model.model.classifier.modules_to_save.default.dense.bias
base_model.model.classifier.modules_to_save.default.out_proj.weight
base_model.model.classifier.modules_to_save.default.out_proj.bias

```
print('PEFT Model')
peft_model.print_trainable_parameters()
```

```
PEFT Model
trainable params: 962,308 || all params: 125,611,016 || trainable%: 0.7661
```

Training Setup

```
# To track evaluation accuracy during training
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_sco

def compute_metrics(pred):
    labels = pred.label_ids
    preds = pred.predictions.argmax(-1)
    # Calculate accuracy
    accuracy = accuracy_score(labels, preds)
    return {
        'accuracy': accuracy
    }
```

```
# Setup Training args
output_dir = "results"
# Better training setup
from transformers import TrainingArguments

training_args = TrainingArguments(
    output_dir="results",
    eval_strategy="epoch",
    save_strategy="epoch",
    learning_rate=2e-5,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=64,
    num_train_epochs=6,
    weight_decay=0.01,
    warmup_ratio=0.1,
    lr_scheduler_type="linear",
    load_best_model_at_end=True,
    metric_for_best_model="accuracy",
    greater_is_better=True,
    fp16=True,
    optim="adamw_torch",
    logging_dir="./logs",
    report_to=None
)
```

```
def get_trainer(model):
      return  Trainer(
          model=model,
          args=training_args,
          compute_metrics=compute_metrics,
          train_dataset=train_dataset,
          eval_dataset=eval_dataset,
          data_collator=data_collator,
      )
```

Start Training

```
peft_lora_finetuning_trainer = get_trainer(peft_model)

result = peft_lora_finetuning_trainer.train()
```

```
No label_names provided for model class `PeftModelForSequenceClassification`.
```

[44760/44760 1:05:05, Epoch 6/6]

Epoch	Training Loss	Validation Loss	Accuracy
1	0.265600	0.324273	0.900000
2	0.245200	0.268220	0.914062
3	0.217600	0.242825	0.928125
4	0.208900	0.234309	0.926562
5	0.202100	0.230531	0.931250
6	0.193800	0.236640	0.925000

Evaluate Finetuned Model

Double-click (or enter) to edit

Performing Inference on Custom Input

Uncomment following functions for running inference on custom inputs

```
def classify(model, tokenizer, text):
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    inputs = tokenizer(text, truncation=True, padding=True, return_tensors="pt")
    output = model(**inputs)

    prediction = output.logits.argmax(dim=-1).item()

    print(f'\n Class: {prediction}, Label: {id2label[prediction]}, Text: {text}'
    return id2label[prediction]


classify( peft_model, tokenizer, "Kederis proclaims innocence Olympic champion K
classify( peft_model, tokenizer, "Wall St. Bears Claw Back Into the Black (Reute
```

```
 Class: 0, Label: World, Text: Kederis proclaims innocence Olympic champion K

 Class: 2, Label: Business, Text: Wall St. Bears Claw Back Into the Black (Re
'Business'
```

Run Inference on eval_dataset

```
from torch.utils.data import DataLoader
import evaluate
from tqdm import tqdm

def evaluate_model(inference_model, dataset, labelled=True, batch_size=8, data_co
    """
    Evaluate a PEFT model on a dataset.

    Args:
        inference_model: The model to evaluate.
        dataset: The dataset (Hugging Face Dataset) to run inference on.
        labelled (bool): If True, the dataset includes labels and metrics will be
                         If False, only predictions will be returned.
        batch_size (int): Batch size for inference.
        data_collator: Function to collate batches. If None, the default collate_

    Returns:
        If labelled is True, returns a tuple (metrics, predictions)
        If labelled is False, returns the predictions.
    """
    # Create the DataLoader
    eval_dataloader = DataLoader(dataset, batch_size=batch_size, collate_fn=data_
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

```
    inference_model.to(device)
    inference_model.eval()

    all_predictions = []
    if labelled:
        metric = evaluate.load('accuracy')

    # Loop over the DataLoader
    for batch in tqdm(eval_dataloader):
        # Move each tensor in the batch to the device
        batch = {k: v.to(device) for k, v in batch.items()}
        with torch.no_grad():
            outputs = inference_model(**batch)
        predictions = outputs.logits.argmax(dim=-1)
        all_predictions.append(predictions.cpu())

        if labelled:
            # Expecting that labels are provided under the "labels" key.
            references = batch["labels"]
            metric.add_batch(
                predictions=predictions.cpu().numpy(),
                references=references.cpu().numpy()
            )

    # Concatenate predictions from all batches
    all_predictions = torch.cat(all_predictions, dim=0)

    if labelled:
        eval_metric = metric.compute()
        print("Evaluation Metric:", eval_metric)
        return eval_metric, all_predictions
    else:
        return all_predictions


# Check evaluation accuracy
_, _ = evaluate_model(peft_model, eval_dataset, True, 8, data_collator)
```

```
100%|██████████| 80/80 [00:01<00:00, 49.58it/s]Evaluation Metric: {'accuracy':
```

```
# Print final evaluation accuracy
final_eval = peft_lora_finetuning_trainer.evaluate()
print(f"\n✅ Final Eval Accuracy: {final_eval['eval_accuracy']:.4f}")

# Show number of parameters
peft_model.print_trainable_parameters()
```

```
[10/10 00:01]

✅ Final Eval Accuracy: 0.9313
trainable params: 962,308 || all params: 125,611,016 || trainable%: 0.7661
```

Start coding or generate with AI.

Run Inference on unlabelled dataset

```
#Load your unlabelled data
unlabelled_dataset = pd.read_pickle("test_unlabelled.pkl")
test_dataset = unlabelled_dataset.map(preprocess, batched=True, remove_columns=["
unlabelled_dataset
```

```
Map: 100%|██████████| 8000/8000 [00:05<00:00, 1429.69 examples/s]
Dataset({
    features: ['text'],
    num_rows: 8000
})
```

```
# Run inference and save predictions
preds = evaluate_model(peft_model, test_dataset, False, 8, data_collator)
df_output = pd.DataFrame({
    'ID': range(len(preds)),
    'Label': preds.numpy()  # or preds.tolist()
})
df_output.to_csv(os.path.join(output_dir,"inference_output.csv"), index=False)
print("Inference complete. Predictions saved to inference_output.csv")
```

```
100%|██████████| 1000/1000 [00:18<00:00, 54.10it/s]
Inference complete. Predictions saved to inference_output.csv
```

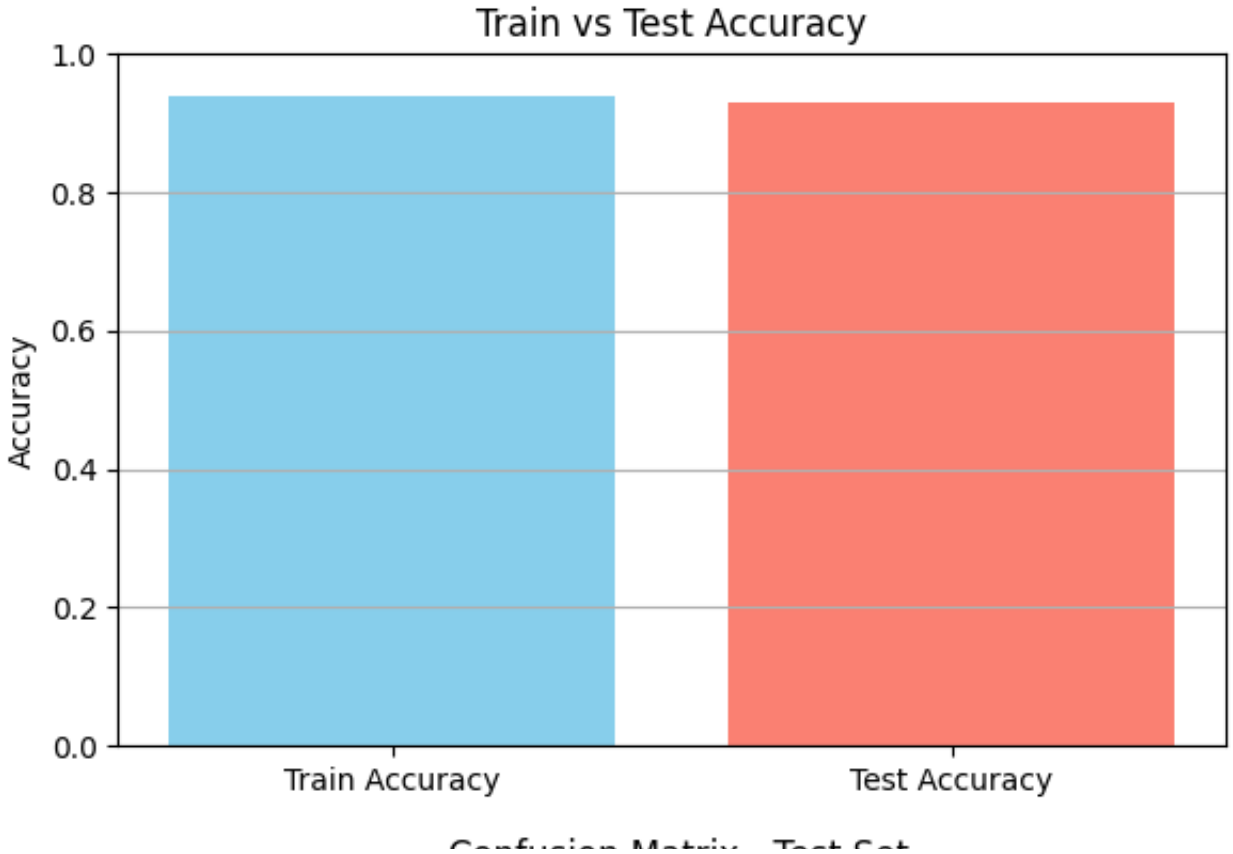
```
# Save model and tokenizer for reproducibility
peft_model.save_pretrained("./results/final_lora_model")
tokenizer.save_pretrained("./results/final_lora_model")
```

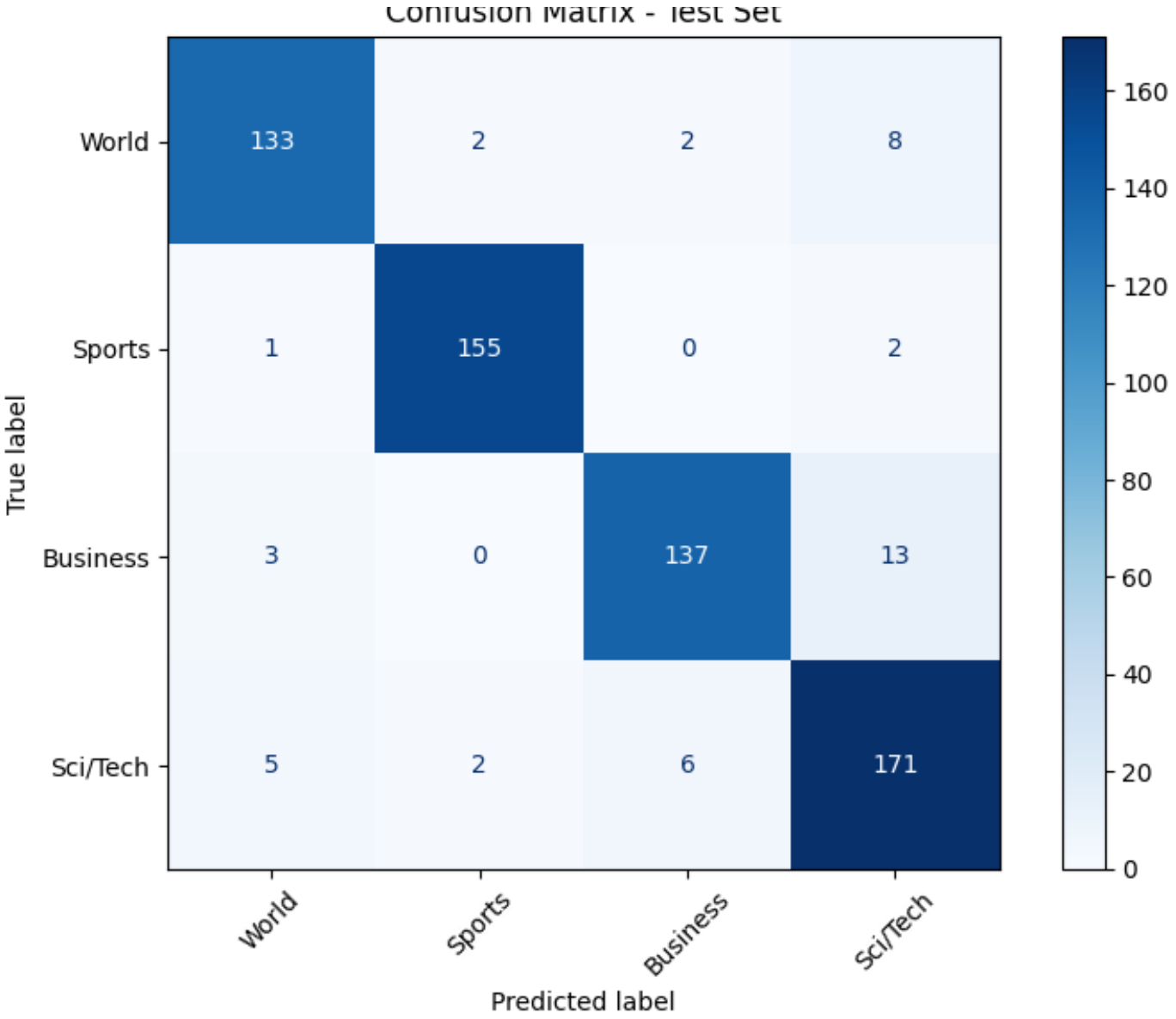
```
('./results/final_lora_model/tokenizer_config.json',
 './results/final_lora_model/special_tokens_map.json',
 './results/final_lora_model/vocab.json',
 './results/final_lora_model/merges.txt',
 './results/final_lora_model/added_tokens.json')
```

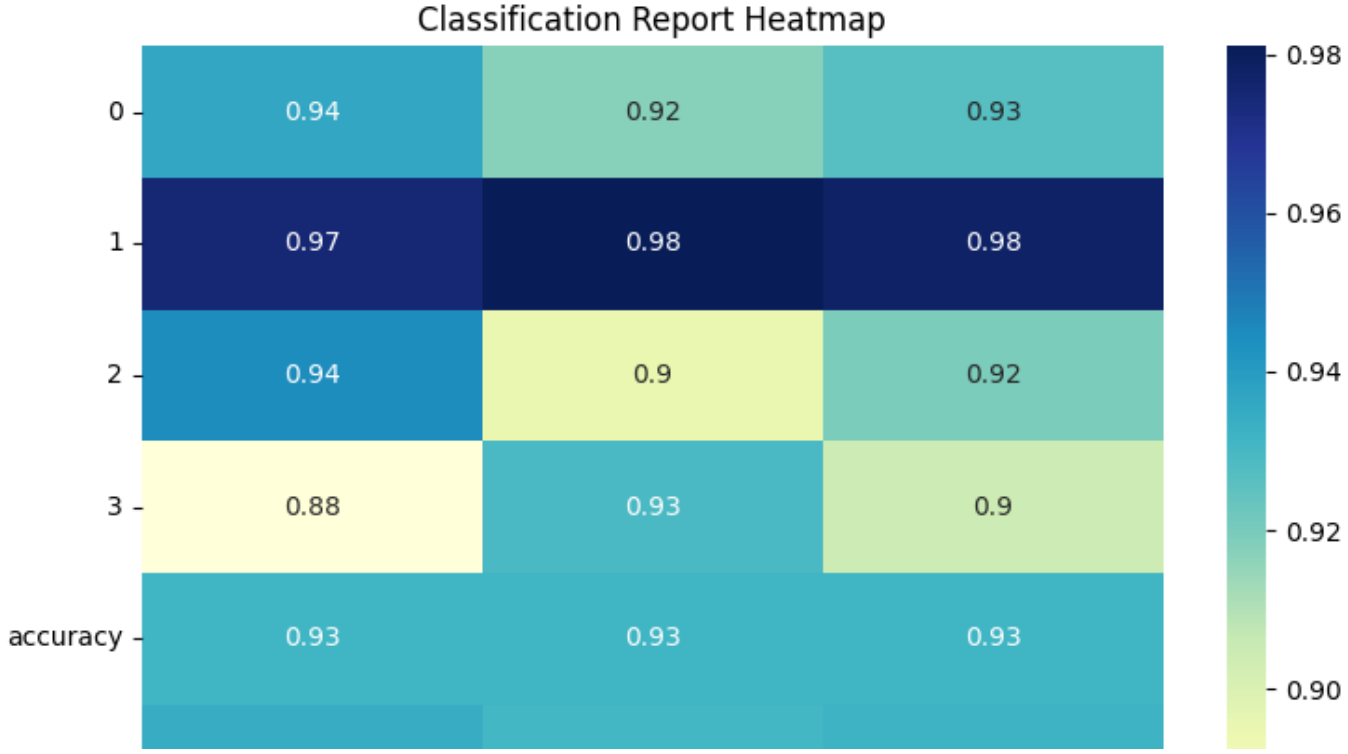
```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_sco

def print_metrics(y_true, y_pred, dataset_name=""):
    acc = accuracy_score(y_true, y_pred)
    prec = precision_score(y_true, y_pred, average='macro', zero_division=0)
    rec = recall_score(y_true, y_pred, average='macro', zero_division=0)
    f1 = f1_score(y_true, y_pred, average='macro', zero_division=0)

    print(f"\n=== {dataset_name} Metrics ===")
    print(classification_report(y_true, y_pred, target_names=class_names, zero_di
    return acc, prec, rec, f1
```

```
# Utility function to extract predictions and labels from a dataset
from torch.utils.data import DataLoader

def get_predictions(model, dataset, batch_size=8, data_collator=None):
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    dataloader = DataLoader(dataset, batch_size=batch_size, collate_fn=data_colla

    model.to(device)
    model.eval()

    all_preds = []
    all_labels = []

    for batch in dataloader:
        labels = batch["labels"].to(device)
        inputs = {k: v.to(device) for k, v in batch.items() if k != "labels"}
        with torch.no_grad():
            outputs = model(**inputs)
        preds = outputs.logits.argmax(dim=-1)

        all_preds.extend(preds.cpu().numpy())
        all_labels.extend(labels.cpu().numpy())

    return all_labels, all_preds



train_labels, train_preds = get_predictions(peft_model, train_dataset, data_colla
test_labels, test_preds = get_predictions(peft_model, eval_dataset, data_collator=




# Run this code after getting predictions to generate plots

from sklearn.metrics import classification_report, confusion_matrix, ConfusionMat
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Accuracy comparison
train_acc = accuracy_score(train_labels, train_preds)
test_acc = accuracy_score(test_labels, test_preds)

plt.figure(figsize=(6, 4))
plt.bar(['Train Accuracy', 'Test Accuracy'], [train_acc, test_acc], color=['skybl
plt.ylim(0, 1)
plt.ylabel('Accuracy')
```

```
plt.title('Train vs Test Accuracy')
plt.grid(axis='y')
plt.tight_layout()
plt.savefig("train_vs_test_accuracy.png")
plt.show()

# Confusion matrix
cm = confusion_matrix(test_labels, test_preds)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=class_names)
fig, ax = plt.subplots(figsize=(8, 6))
disp.plot(cmap='Blues', xticks_rotation=45, ax=ax)
plt.title("Confusion Matrix - Test Set")
plt.tight_layout()
plt.savefig("confusion_matrix_test_set.png")
plt.show()

# Classification report heatmap
report = classification_report(test_labels, test_preds, output_dict=True)
df_report = pd.DataFrame(report).iloc[:-1, :-1].T
plt.figure(figsize=(8, 5))
sns.heatmap(df_report, annot=True, cmap="YlGnBu")
plt.title("Classification Report Heatmap")
plt.tight_layout()
plt.savefig("classification_report_heatmap.png")
plt.show()
```




Confusion Matrix - Test Set
World
Sports
Business
Sci/Tech
True label
133
2
2
8
1
155
0
2
3
0
137
13
5
2
6
171
160
140
120
100
80
60
40
20
0
World
Sports
Business
Sci/Tech
Predicted label


Classification Report Heatmap
0
1
2
3
accuracy
0.94
0.92
0.93
0.97
0.98
0.98
0.94
0.9
0.92
0.88
0.93
0.9
0.93
0.93
0.93
0.98
0.96
0.94
0.92
0.90


macro avg
0.93
0.93
0.93
precision
recall
f1-score

```
!pip install matplotlib
```

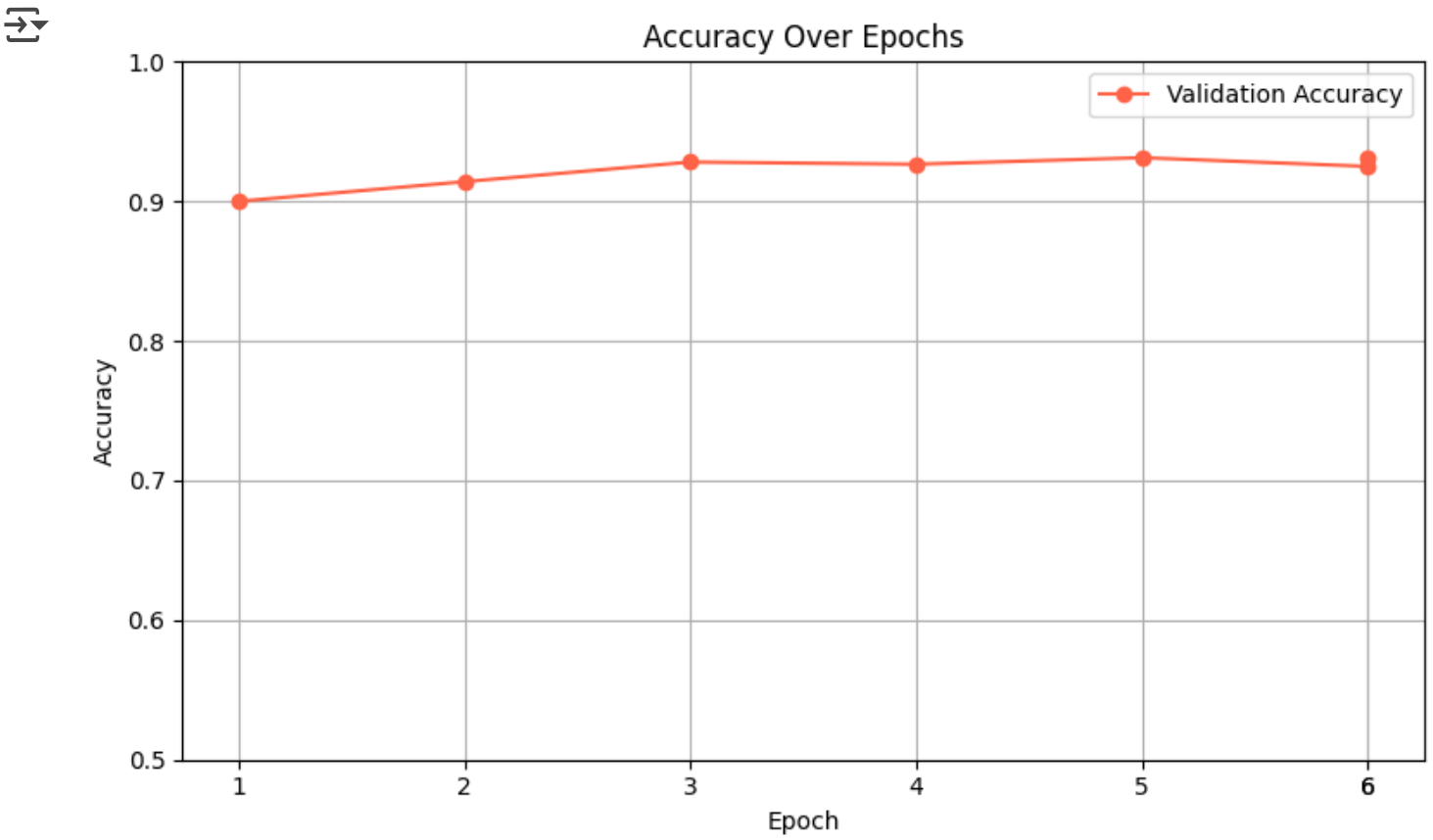
```
Defaulting to user installation because normal site-packages is not writeable
Collecting matplotlib
  Downloading matplotlib-3.9.4-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x
Collecting contourpy>=1.0.1 (from matplotlib)
  Downloading contourpy-1.3.0-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x8
Collecting cycler>=0.10 (from matplotlib)
  Downloading cycler-0.12.1-py3-none-any.whl.metadata (3.8 kB)
Collecting fonttools>=4.22.0 (from matplotlib)
  Downloading fonttools-4.57.0-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 102.5/102.5 kB 1.3 MB/s eta 0:00:
Collecting kiwisolver>=1.3.1 (from matplotlib)
  Downloading kiwisolver-1.4.7-cp39-cp39-manylinux_2_12_x86_64.manylinux2010_x
Requirement already satisfied: numpy>=1.23 in /home/ab12660/.local/lib/python3
Requirement already satisfied: packaging>=20.0 in /share/apps/pyenv/py3.9/lib/
Collecting pillow>=8 (from matplotlib)
  Downloading pillow-11.2.1-cp39-cp39-manylinux_2_28_x86_64.whl.metadata (8.9
Collecting pyparsing>=2.3.1 (from matplotlib)
  Downloading pyparsing-3.2.3-py3-none-any.whl.metadata (5.0 kB)
Requirement already satisfied: python-dateutil>=2.7 in /share/apps/pyenv/py3.9
Collecting importlib-resources>=3.2.0 (from matplotlib)
  Downloading importlib_resources-6.5.2-py3-none-any.whl.metadata (3.9 kB)
Requirement already satisfied: zipp>=3.1.0 in /share/apps/pyenv/py3.9/lib/pyth
Requirement already satisfied: six>=1.5 in /share/apps/pyenv/py3.9/lib/python3
Downloading matplotlib-3.9.4-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.3/8.3 MB 48.1 MB/s eta 0:00:00:0
Downloading contourpy-1.3.0-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 321.9/321.9 kB 8.8 MB/s eta 0:00:0
Downloading cycler-0.12.1-py3-none-any.whl (8.3 kB)
Downloading fonttools-4.57.0-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.6/4.6 MB 53.3 MB/s eta 0:00:00:0
Downloading importlib_resources-6.5.2-py3-none-any.whl (37 kB)
Downloading kiwisolver-1.4.7-cp39-cp39-manylinux_2_12_x86_64.manylinux2010_x86
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.6/1.6 MB 32.3 MB/s eta 0:00:00:0
Downloading pillow-11.2.1-cp39-cp39-manylinux_2_28_x86_64.whl (4.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.6/4.6 MB 46.6 MB/s eta 0:00:00:0
Downloading pyparsing-3.2.3-py3-none-any.whl (111 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 111.1/111.1 kB 2.8 MB/s eta 0:00:0
Installing collected packages: pyparsing, pillow, kiwisolver, importlib-resou
  WARNING: The scripts fonttools, pyftmerge, pyftsubset and ttx are installed
  Consider adding this directory to PATH or, if you prefer to suppress this wa
Successfully installed contourpy-1.3.0 cycler-0.12.1 fonttools-4.57.0 importli
```

```
!pip install seaborn
```

```
Defaulting to user installation because normal site-packages is not writeable
Collecting seaborn
  Downloading seaborn-0.13.2-py3-none-any.whl.metadata (5.4 kB)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in /home/ab12660/.local/li
Requirement already satisfied: pandas>=1.2 in /home/ab12660/.local/lib/python3
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in /home/ab12660/.local
Requirement already satisfied: contourpy>=1.0.1 in /home/ab12660/.local/lib/py
Requirement already satisfied: cycler>=0.10 in /home/ab12660/.local/lib/python
Requirement already satisfied: fonttools>=4.22.0 in /home/ab12660/.local/lib/p
Requirement already satisfied: kiwisolver>=1.3.1 in /home/ab12660/.local/lib/p
Requirement already satisfied: packaging>=20.0 in /share/apps/pyenv/py3.9/lib/
Requirement already satisfied: pillow>=8 in /home/ab12660/.local/lib/python3.9
Requirement already satisfied: pyparsing>=2.3.1 in /home/ab12660/.local/lib/py
Requirement already satisfied: python-dateutil>=2.7 in /share/apps/pyenv/py3.9
Requirement already satisfied: importlib-resources>=3.2.0 in /home/ab12660/.lo
Requirement already satisfied: pytz>=2020.1 in /home/ab12660/.local/lib/python
Requirement already satisfied: tzdata>=2022.7 in /home/ab12660/.local/lib/pyth
Requirement already satisfied: zipp>=3.1.0 in /share/apps/pyenv/py3.9/lib/pyth
Requirement already satisfied: six>=1.5 in /share/apps/pyenv/py3.9/lib/python3
Downloading seaborn-0.13.2-py3-none-any.whl (294 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 294.9/294.9 kB 3.2 MB/s eta 0:00:0
Installing collected packages: seaborn
Successfully installed seaborn-0.13.2
```

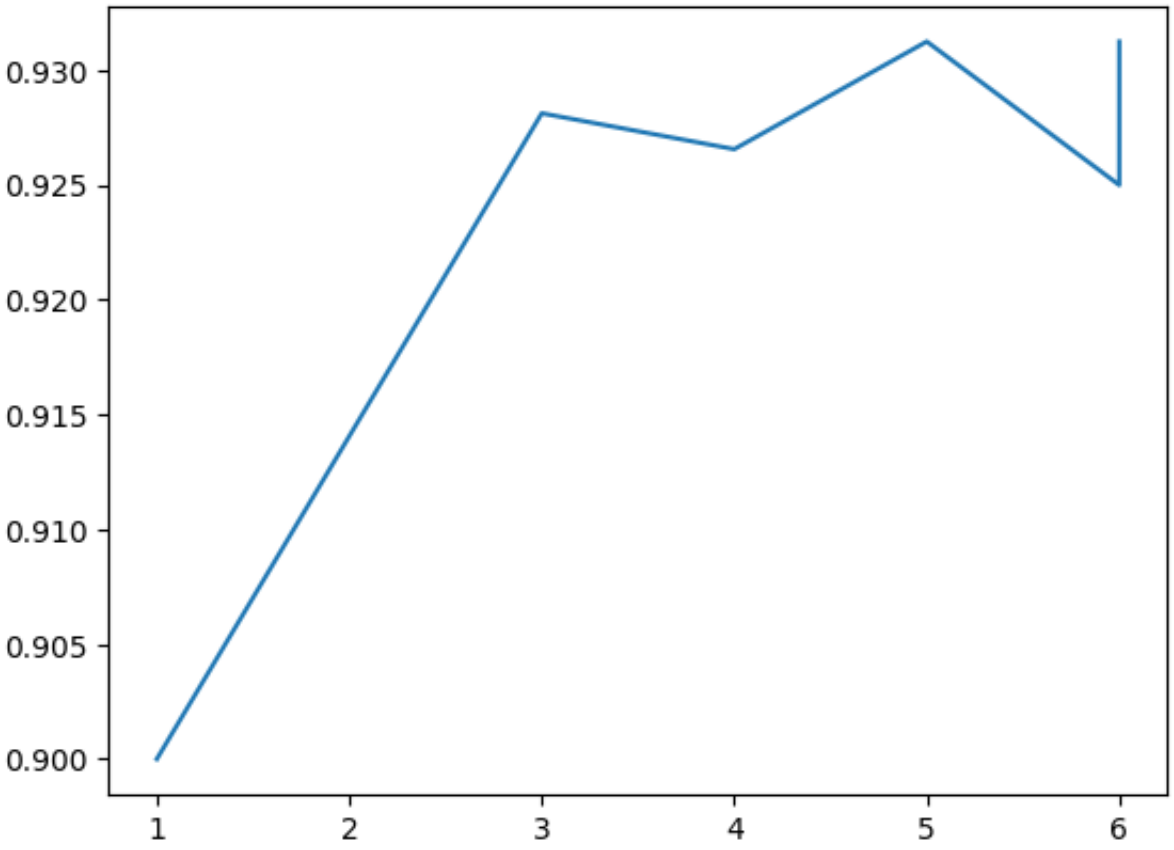
```
# Get the logs stored by the Trainer
log_history = peft_lora_finetuning_trainer.state.log_history

# Extract accuracy values per epoch
train_accs = []
eval_accs = []
epochs = []

for entry in log_history:
    if 'eval_accuracy' in entry:
        eval_accs.append(entry['eval_accuracy'])
        epochs.append(entry['epoch'])
    if 'train_accuracy' in entry:  # Only if training accuracy is logged
        train_accs.append(entry['train_accuracy'])  # Optional; might be missing

# Optional: manually insert training accuracy if it's missing
if not train_accs:
    train_accs = [None] * len(eval_accs)
```

```
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 5))
plt.plot(epochs, eval_accs, label='Validation Accuracy', marker='o', color='tomat
if any(train_accs):
    plt.plot(epochs, train_accs, label='Training Accuracy', marker='o', color='sk

plt.title("Accuracy Over Epochs")
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.ylim(0.5, 1.0)
plt.xticks(epochs)
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.savefig("accuracy_vs_epoch.png")  # For your report
plt.show()
```

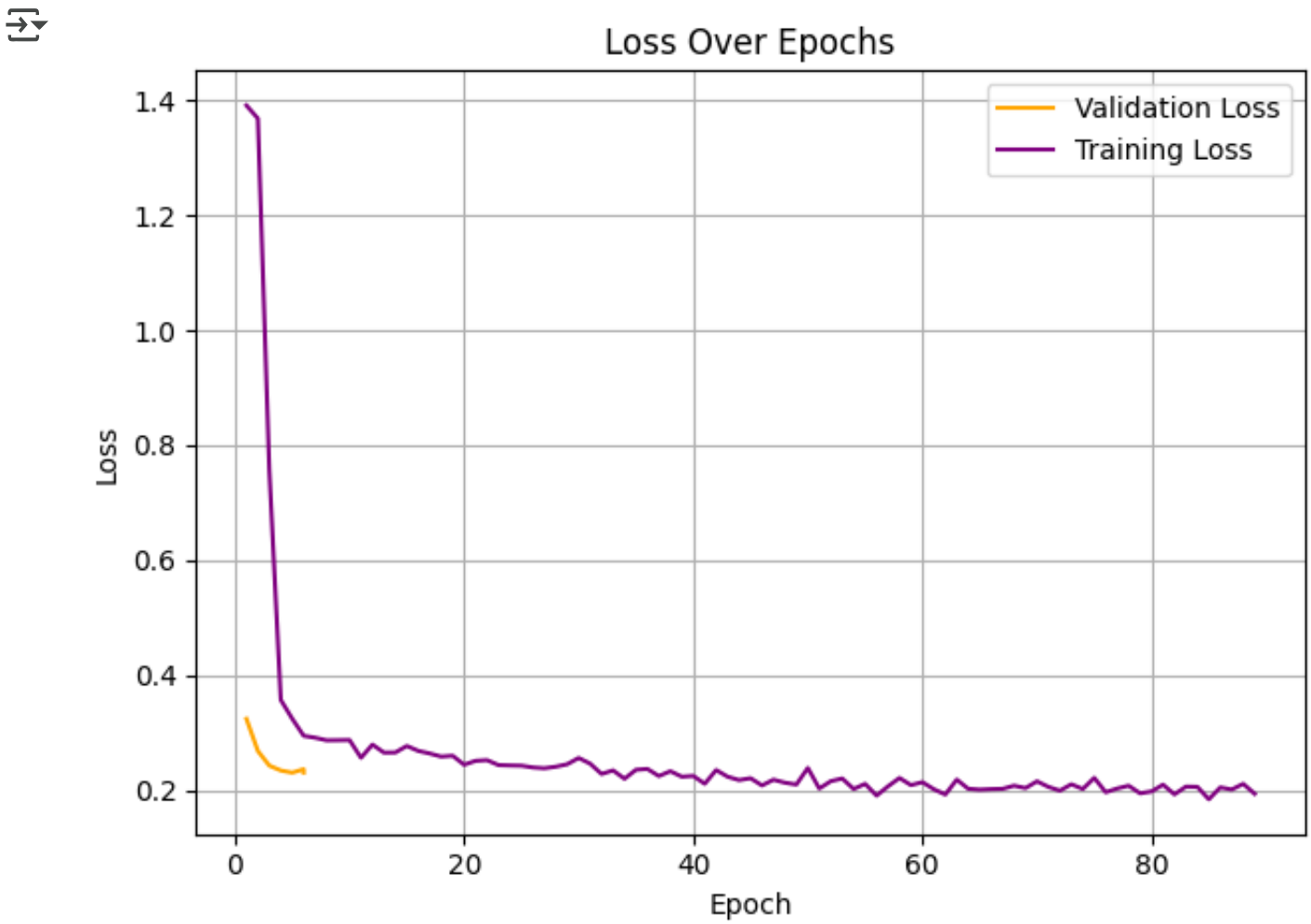


```
plt.plot(epochs, eval_accs, label='Validation Accuracy')
```

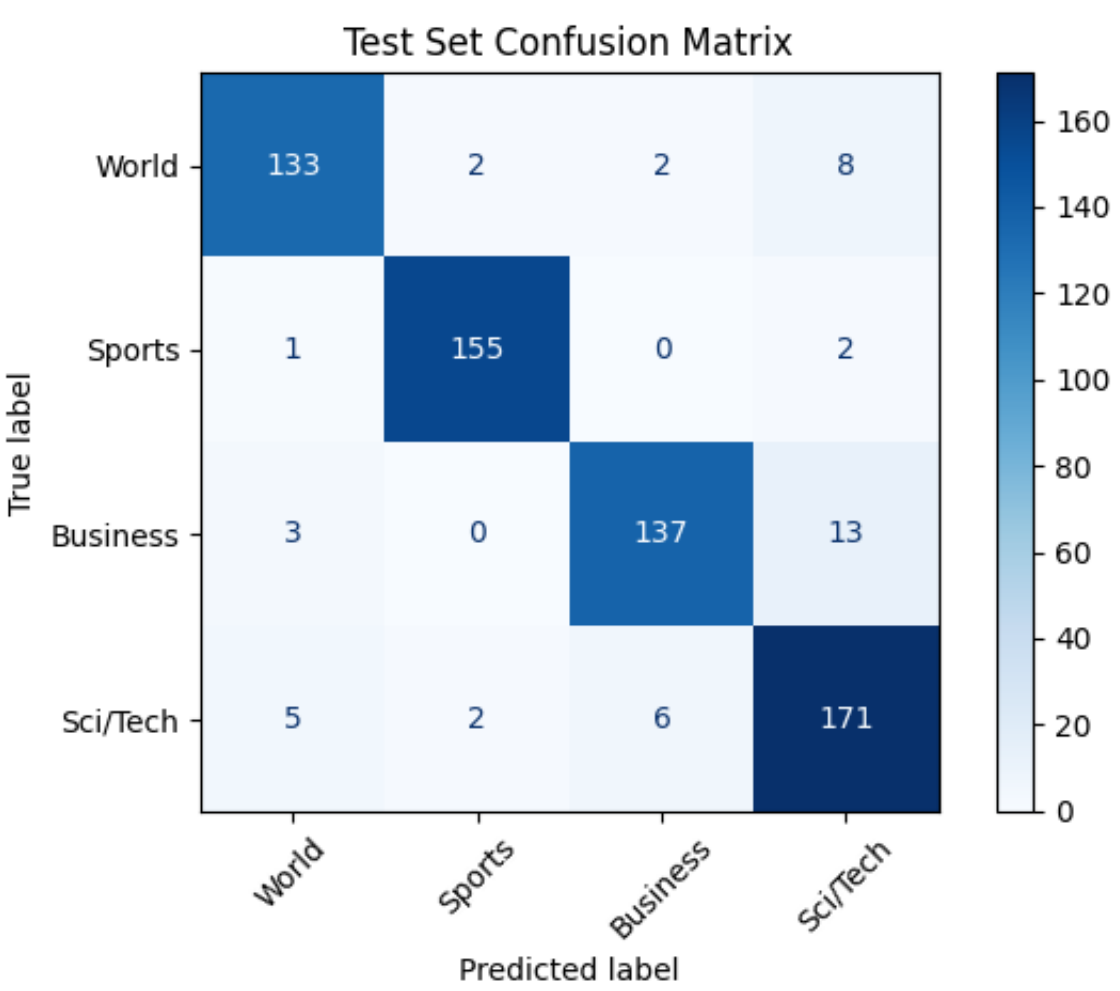
```
[<matplotlib.lines.Line2D at 0x14e5651a5bb0>]
```



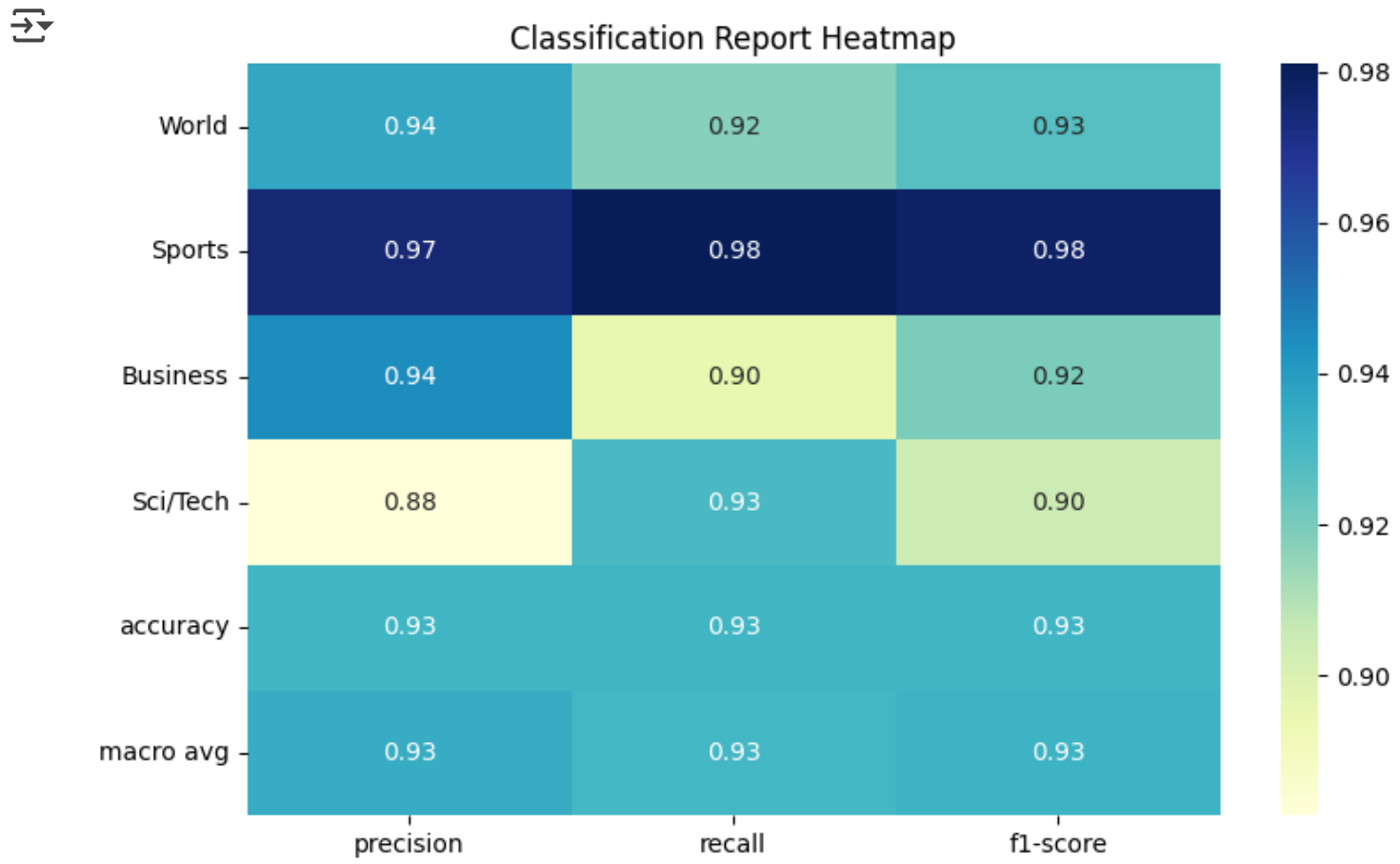
```
train_losses = [entry['loss'] for entry in log_history if 'loss' in entry]
eval_losses = [entry['eval_loss'] for entry in log_history if 'eval_loss' in entr
epochs_loss = [entry['epoch'] for entry in log_history if 'eval_loss' in entry]

plt.plot(epochs_loss, eval_losses, label='Validation Loss', color='orange')
plt.plot(range(1, len(train_losses)+1), train_losses, label='Training Loss', colo
plt.title("Loss Over Epochs")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.savefig("loss_vs_epoch.png")
plt.show()
```



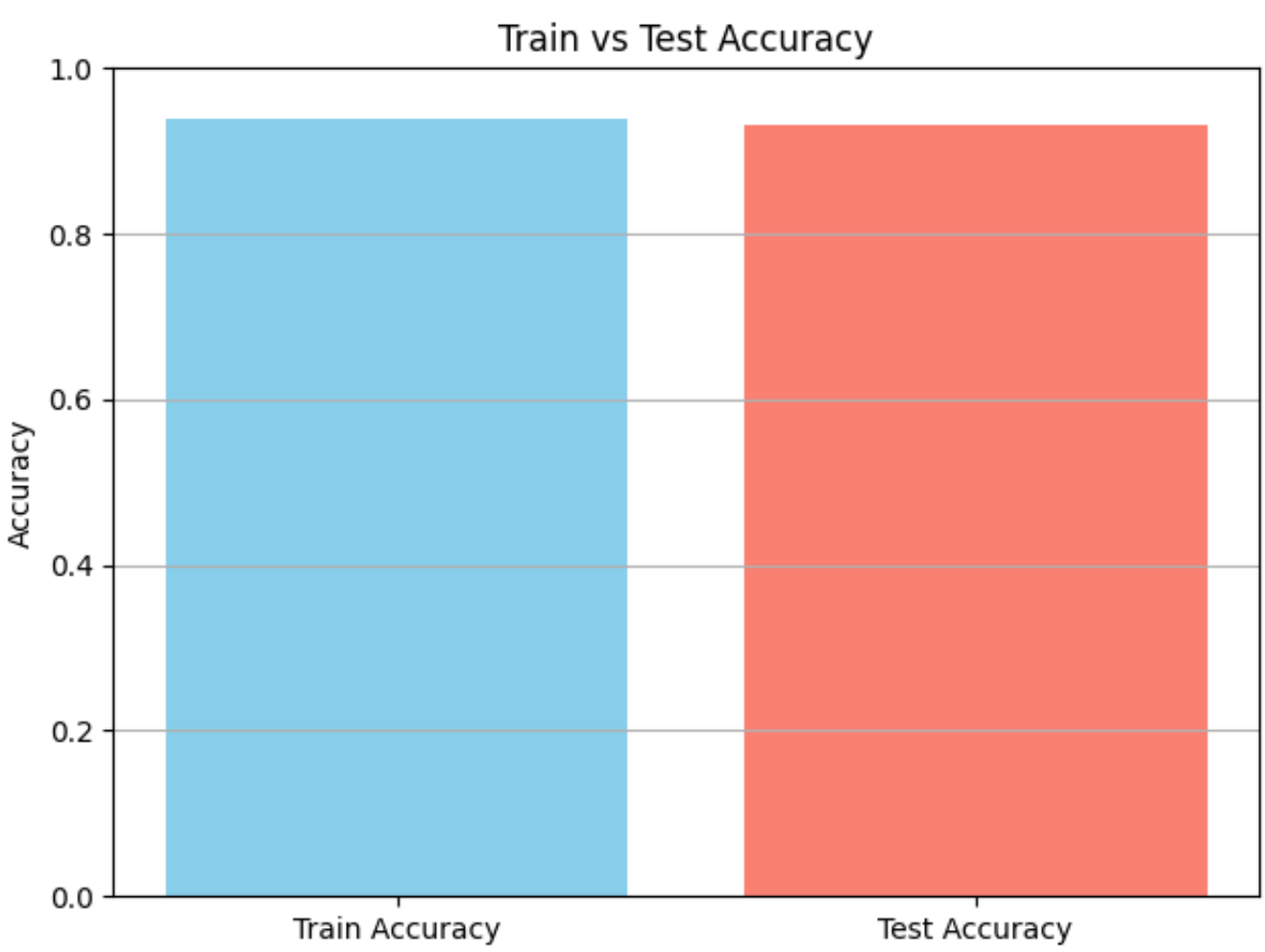
```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

cm = confusion_matrix(test_labels, test_preds)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=class_names)
disp.plot(cmap='Blues', xticks_rotation=45)
plt.title("Test Set Confusion Matrix")
plt.tight_layout()
plt.savefig("confusion_matrix.png")
plt.show()
```



```
from sklearn.metrics import classification_report

report = classification_report(test_labels, test_preds, target_names=class_names,
df_report = pd.DataFrame(report).transpose()
df_report.to_csv("classification_report.csv")

# Optional visualization:
import seaborn as sns
plt.figure(figsize=(8, 5))
sns.heatmap(df_report.iloc[:-1, :-1], annot=True, cmap="YlGnBu", fmt=".2f")
plt.title("Classification Report Heatmap")
plt.tight_layout()
plt.savefig("classification_report_heatmap.png")
plt.show()
```



```
plt.bar(['Train Accuracy', 'Test Accuracy'], [train_acc, test_acc], color=['skybl
plt.ylim(0, 1)
plt.ylabel('Accuracy')
plt.title('Train vs Test Accuracy')
plt.grid(axis='y')
plt.tight_layout()
plt.savefig("train_vs_test_accuracy.png")
plt.show()
```



```
examples = [
    "NASA launches a new satellite into orbit.",
    "Stocks rally as Fed pauses interest rate hikes.",
    "Manchester United defeats Liverpool in derby match.",
    "New AI model breaks records in image generation."
]

for sentence in examples:
    pred = classify(peft_model, tokenizer, sentence)
    print(f"Text: {sentence}\n → Predicted: {pred}\n")
```

```
 Class: 3, Label: Sci/Tech, Text: NASA launches a new satellite into orbit.
Text: NASA launches a new satellite into orbit.
 → Predicted: Sci/Tech


 Class: 2, Label: Business, Text: Stocks rally as Fed pauses interest rate hil
Text: Stocks rally as Fed pauses interest rate hikes.
 → Predicted: Business


 Class: 1, Label: Sports, Text: Manchester United defeats Liverpool in derby r
Text: Manchester United defeats Liverpool in derby match.
 → Predicted: Sports


 Class: 3, Label: Sci/Tech, Text: New AI model breaks records in image generat
Text: New AI model breaks records in image generation.
 → Predicted: Sci/Tech
```

Start coding or generate with AI.