In [1]:
```
!pip install -q peft transformers datasets evaluate
```

In [2]:
```
from huggingface_hub import notebook_login

notebook_login()
```

```
VBox(children=(HTML(value='<center> <img\nsrc=https://huggingface.co/front/assets/huggingface_logo-noborder.sv…
```

In [3]:
```
from transformers import (
    AutoModelForSequenceClassification,
    AutoTokenizer,
    DataCollatorWithPadding,
    TrainingArguments,
    Trainer,
)
from peft import (
    get_peft_config,
    get_peft_model,
    get_peft_model_state_dict,
    set_peft_model_state_dict,
    PeftType,
    PromptEncoderConfig,
)
from datasets import load_dataset
import evaluate
import torch

model_name_or_path = "roberta-large"
task = "mrpc"
num_epochs = 20
lr = 1e-3
batch_size = 32
```

```
2024-02-22 07:19:21.200056: E external/local_xla/xla/stream_executor/cuda/cuda_dnn.cc:9261] Unable to register
cuDNN factory: Attempting to register factory for plugin cuDNN when one has already been registered
2024-02-22 07:19:21.200183: E external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:607] Unable to register c
uFFT factory: Attempting to register factory for plugin cuFFT when one has already been registered
2024-02-22 07:19:21.359845: E external/local_xla/xla/stream_executor/cuda/cuda_blas.cc:1515] Unable to register
cuBLAS factory: Attempting to register factory for plugin cuBLAS when one has already been registered
```

In [4]:
```
dataset = load_dataset("glue", task)
dataset["train"][0]
```

```
Downloading builder script:   0%|          | 0.00/7.78k [00:00<?, ?B/s]
Downloading metadata:   0%|          | 0.00/4.47k [00:00<?, ?B/s]
Downloading and preparing dataset glue/mrpc (download: 1.43 MiB, generated: 1.43 MiB, post-processed: Unknown s
ize, total: 2.85 MiB) to /root/.cache/huggingface/datasets/glue/mrpc/1.0.0/dacbe3125aa31d7f70367a07a8a9e72a5a0b
feb5fc42e75c9db75b96da6053ad...
Downloading data files:   0%|          | 0/3 [00:00<?, ?it/s]
Downloading data: 0.00B [00:00, ?B/s]
Downloading data: 0.00B [00:00, ?B/s]
Downloading data: 0.00B [00:00, ?B/s]
Generating train split:   0%|          | 0/3668 [00:00<?, ? examples/s]
Generating validation split:   0%|          | 0/408 [00:00<?, ? examples/s]
Generating test split:   0%|          | 0/1725 [00:00<?, ? examples/s]
Dataset glue downloaded and prepared to /root/.cache/huggingface/datasets/glue/mrpc/1.0.0/dacbe3125aa31d7f70367
a07a8a9e72a5a0bfeb5fc42e75c9db75b96da6053ad. Subsequent calls will reuse this data.
  0%|          | 0/3 [00:00<?, ?it/s]
```

Out[4]:
```
{'sentence1': 'Amrozi accused his brother , whom he called " the witness " , of deliberately distorting his evi
dence .',
 'sentence2': 'Referring to him as only " the witness " , Amrozi accused his brother of deliberately distorting
his evidence .',
 'label': 1,
 'idx': 0}
```

In [5]:
```
metric = evaluate.load("glue", task)
```

```
Downloading builder script:   0%|          | 0.00/5.75k [00:00<?, ?B/s]
```

In [6]:
```
import numpy as np


def compute_metrics(eval_pred):
    predictions, labels = eval_pred
    predictions = np.argmax(predictions, axis=1)
    return metric.compute(predictions=predictions, references=labels)
```

In [7]:
```
if any(k in model_name_or_path for k in ("gpt", "opt", "bloom")):
    padding_side = "left"
else:
    padding_side = "right"

tokenizer = AutoTokenizer.from_pretrained(model_name_or_path, padding_side=padding_side)
if getattr(tokenizer, "pad_token_id") is None:
    tokenizer.pad_token_id = tokenizer.eos_token_id
```

```
def tokenize_function(examples):
    # max_length=None => use the model max length (it's actually the default)
    outputs = tokenizer(examples["sentence1"], examples["sentence2"], truncation=True, max_length=None)
    return outputs
```

```
tokenizer_config.json:   0%|          | 0.00/25.0 [00:00<?, ?B/s]
config.json:   0%|          | 0.00/482 [00:00<?, ?B/s]
vocab.json:   0%|          | 0.00/899k [00:00<?, ?B/s]
merges.txt:   0%|          | 0.00/456k [00:00<?, ?B/s]
tokenizer.json:   0%|          | 0.00/1.36M [00:00<?, ?B/s]
```

In [8]:
```
tokenized_datasets = dataset.map(
    tokenize_function,
    batched=True,
    remove_columns=["idx", "sentence1", "sentence2"],
)

tokenized_datasets = tokenized_datasets.rename_column("label", "labels")
```

```
  0%|          | 0/4 [00:00<?, ?ba/s]
  0%|          | 0/1 [00:00<?, ?ba/s]
  0%|          | 0/2 [00:00<?, ?ba/s]
```

In [9]:
```
data_collator = DataCollatorWithPadding(tokenizer=tokenizer, padding="longest")
```

In [10]:
```
peft_config = PromptEncoderConfig(task_type="SEQ_CLS", num_virtual_tokens=20, encoder_hidden_size=128)
```

In [11]:
```
model = AutoModelForSequenceClassification.from_pretrained(model_name_or_path, return_dict=True)
model = get_peft_model(model, peft_config)
model.print_trainable_parameters()
```

```
model.safetensors:   0%|          | 0.00/1.42G [00:00<?, ?B/s]
Some weights of RobertaForSequenceClassification were not initialized from the model checkpoint at roberta-large and are newly initialized: ['classifier.dense.bias', 'classifier.dense.weight', 'classifier.out_proj.bias', 'classifier.out_proj.weight']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
trainable params: 1,351,938 || all params: 356,713,732 || trainable%: 0.3789980252288129
```

In [14]:
```
training_args = TrainingArguments(
    output_dir="outputs/roberta-large-peft-p-tuning",
    learning_rate=1e-3,
    per_device_train_batch_size=32,
    per_device_eval_batch_size=32,
    num_train_epochs=10,
    weight_decay=0.01,
    evaluation_strategy="epoch",
    save_strategy="epoch",
    load_best_model_at_end=True,
)
```

In [15]:
```
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_datasets["train"],
    eval_dataset=tokenized_datasets["test"],
    tokenizer=tokenizer,
    data_collator=data_collator,
    compute_metrics=compute_metrics,
)

trainer.train()
```

[580/580 17:44, Epoch 10/10]

Epoch	Training Loss	Validation Loss	Accuracy	F1
1	No log	0.609398	0.679420	0.801579
2	No log	0.666898	0.673043	0.801966
3	No log	0.614854	0.681739	0.804278
4	No log	0.690202	0.665507	0.799025
5	No log	0.590705	0.696812	0.806368
6	No log	0.615271	0.680580	0.802580
7	No log	0.594967	0.693913	0.807860
8	No log	0.605051	0.689855	0.807346
9	0.620900	0.591118	0.697391	0.808931
10	0.620900	0.591311	0.697391	0.808229

```
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/_functions.py:68: UserWarning: Was asked to gather al
ong dimension 0, but all input tensors were scalars; will instead unsqueeze and return a vector.
  warnings.warn('Was asked to gather along dimension 0, but all '
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/_functions.py:68: UserWarning: Was asked to gather al
ong dimension 0, but all input tensors were scalars; will instead unsqueeze and return a vector.
  warnings.warn('Was asked to gather along dimension 0, but all '
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/_functions.py:68: UserWarning: Was asked to gather al
ong dimension 0, but all input tensors were scalars; will instead unsqueeze and return a vector.
  warnings.warn('Was asked to gather along dimension 0, but all '
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/_functions.py:68: UserWarning: Was asked to gather al
ong dimension 0, but all input tensors were scalars; will instead unsqueeze and return a vector.
  warnings.warn('Was asked to gather along dimension 0, but all '
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/_functions.py:68: UserWarning: Was asked to gather al
ong dimension 0, but all input tensors were scalars; will instead unsqueeze and return a vector.
  warnings.warn('Was asked to gather along dimension 0, but all '
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/_functions.py:68: UserWarning: Was asked to gather al
ong dimension 0, but all input tensors were scalars; will instead unsqueeze and return a vector.
  warnings.warn('Was asked to gather along dimension 0, but all '
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/_functions.py:68: UserWarning: Was asked to gather al
ong dimension 0, but all input tensors were scalars; will instead unsqueeze and return a vector.
  warnings.warn('Was asked to gather along dimension 0, but all '
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/_functions.py:68: UserWarning: Was asked to gather al
ong dimension 0, but all input tensors were scalars; will instead unsqueeze and return a vector.
  warnings.warn('Was asked to gather along dimension 0, but all '
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/_functions.py:68: UserWarning: Was asked to gather al
ong dimension 0, but all input tensors were scalars; will instead unsqueeze and return a vector.
  warnings.warn('Was asked to gather along dimension 0, but all '
```

Out[15]:

```
TrainOutput(global_step=580, training_loss=0.6159444874730603, metrics={'train_runtime': 1065.3397, 'train_samp
les_per_second': 34.43, 'train_steps_per_second': 0.544, 'total_flos': 5639290068053376.0, 'train_loss': 0.6159
444874730603, 'epoch': 10.0})
```

In [16]:

```python
model.push_to_hub("likhith231/roberta-large-peft-p-tuning", use_auth_token=True)
```

```
/opt/conda/lib/python3.10/site-packages/transformers/utils/hub.py:821: FutureWarning: The `use_auth_token` argu
ment is deprecated and will be removed in v5 of Transformers. Please use `token` instead.
  warnings.warn(
adapter_model.safetensors:   0%|           | 0.00/4.29M [00:00<?, ?B/s]
```

Out[16]:

```
CommitInfo(commit_url='https://huggingface.co/likhith231/roberta-large-peft-p-tuning/commit/e61b2329a53dea2a09d
24dcafa5af3fb0b1d4b77', commit_message='Upload model', commit_description='', oid='e61b2329a53dea2a09d24dcafa5a
f3fb0b1d4b77', pr_url=None, pr_revision=None, pr_num=None)
```

In [ ]:

```python
import torch
from peft import PeftModel, PeftConfig
from transformers import AutoModelForSequenceClassification, AutoTokenizer

peft_model_id = "likhith231/roberta-large-peft-p-tuning"
config = PeftConfig.from_pretrained(peft_model_id)
inference_model = AutoModelForSequenceClassification.from_pretrained(config.base_model_name_or_path)
tokenizer = AutoTokenizer.from_pretrained(config.base_model_name_or_path)
model = PeftModel.from_pretrained(inference_model, peft_model_id)
```

In [18]:

```python
classes = ["not equivalent", "equivalent"]

sentence1 = "Coast redwood trees are the tallest trees on the planet and can grow over 300 feet tall."
sentence2 = "The coast redwood trees, which can attain a height of over 300 feet, are the tallest trees on eart

inputs = tokenizer(sentence1, sentence2, truncation=True, padding="longest", return_tensors="pt")

with torch.no_grad():
    outputs = model(**inputs).logits
    print(outputs)

paraphrased_text = torch.softmax(outputs, dim=1).tolist()[0]
for i in range(len(classes)):
    print(f"{classes[i]}: {int(round(paraphrased_text[i] * 100))}%")
```

```
tensor([[-0.2172,  0.1739]])
not equivalent: 40%
equivalent: 60%
```