```
# Import necessary libraries
import os

# Define file paths (update if necessary)
buggy_file = "/content/train.buggy-fixed.buggy"  # Path to buggy code file
fixed_file = "/content/train.buggy-fixed.fixed"  # Path to fixed code file

# Read and display a few examples
with open(buggy_file, "r", encoding="utf-8") as f_buggy, open(fixed_file, "r", encoding="utf
    buggy_lines = f_buggy.readlines()
    fixed_lines = f_fixed.readlines()

print(len(buggy_lines))
print(len(fixed_lines))
```

```
52364
52364
```

```
import os

def load_data(buggy_file, fixed_file):
    with open(buggy_file, 'r', encoding='utf-8') as buggy_f, open(fixed_file, 'r', encoding=
        buggy_lines = buggy_f.readlines()  # Corrected variable name
        fixed_lines = fixed_f.readlines()  # Corrected variable name

    print(f"🔹 Buggy Samples: {len(buggy_lines)} | Fixed Samples: {len(fixed_lines)}")  # D

    assert len(buggy_lines) == len(fixed_lines), "Mismatch between buggy and fixed samples!"

    return buggy_lines, fixed_lines


def preprocess_code(code):
    """
    Basic preprocessing: Removes extra spaces, converts tabs to spaces, and normalizes Java
    """
    code = code.replace("\t", " ")  # Convert tabs to spaces
    code = "\n".join([line.strip() for line in code.split("\n") if line.strip()])  # Trim ea
    return code

def save_preprocessed_data(buggy_codes, fixed_codes, output_file):
    """
    Saves preprocessed buggy-fixed code pairs to a structured text file.
    """
    with open(output_file, 'w', encoding='utf-8') as f:
        for buggy, fixed in zip(buggy_codes, fixed_codes):
            f.write("BUGGY:\n" + buggy + "\nFIXED:\n" + fixed + "\n=====\n")

# File paths
```

```
dataset_path = "/content"
train_buggy = os.path.join(dataset_path, "train.buggy-fixed.buggy")
train_fixed = os.path.join(dataset_path, "train.buggy-fixed.fixed")

# Load and preprocess data
buggy_samples, fixed_samples = load_data(train_buggy, train_fixed)
buggy_samples = [preprocess_code(code) for code in buggy_samples]
fixed_samples = [preprocess_code(code) for code in fixed_samples]

# Save cleaned data
save_preprocessed_data(buggy_samples, fixed_samples, "preprocessed_train.txt")

print(f"✅ Preprocessing complete! Saved {len(buggy_samples)} samples to 'preprocessed_trai
```

```
🔹 Buggy Samples: 52364 | Fixed Samples: 52364
✅ Preprocessing complete! Saved 52364 samples to 'preprocessed_train.txt'.
```

```
!pip install transformers accelerate bitsandbytes peft torch
```

```
Requirement already satisfied: transformers in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: accelerate in /usr/local/lib/python3.11/dist-packages
Collecting bitsandbytes
  Downloading bitsandbytes-0.45.4-py3-none-manylinux_2_24_x86_64.whl.metadata (5.0 kB
Requirement already satisfied: peft in /usr/local/lib/python3.11/dist-packages (0.14.
Requirement already satisfied: torch in /usr/local/lib/python3.11/dist-packages (2.6.
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (f
Requirement already satisfied: huggingface-hub<1.0,>=0.26.0 in /usr/local/lib/python3
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (f
Requirement already satisfied: tokenizers<0.22,>=0.21 in /usr/local/lib/python3.11/di
Requirement already satisfied: safetensors>=0.4.3 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: psutil in /usr/local/lib/python3.11/dist-packages (fro
Requirement already satisfied: typing-extensions>=4.10.0 in /usr/local/lib/python3.11
Requirement already satisfied: networkx in /usr/local/lib/python3.11/dist-packages (f
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (fro
Requirement already satisfied: fsspec in /usr/local/lib/python3.11/dist-packages (fro
Collecting nvidia-cuda-nvrtc-cu12==12.4.127 (from torch)
  Downloading nvidia_cuda_nvrtc_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metad
Collecting nvidia-cuda-runtime-cu12==12.4.127 (from torch)
  Downloading nvidia_cuda_runtime_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.met
Collecting nvidia-cuda-cupti-cu12==12.4.127 (from torch)
  Downloading nvidia_cuda_cupti_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metad
Collecting nvidia-cudnn-cu12==9.1.0.70 (from torch)
  Downloading nvidia_cudnn_cu12-9.1.0.70-py3-none-manylinux2014_x86_64.whl.metadata (
Collecting nvidia-cublas-cu12==12.4.5.8 (from torch)
  Downloading nvidia_cublas_cu12-12.4.5.8-py3-none-manylinux2014_x86_64.whl.metadata
Collecting nvidia-cufft-cu12==11.2.1.3 (from torch)
  Downloading nvidia_cufft_cu12-11.2.1.3-py3-none-manylinux2014_x86_64.whl.metadata (
```

```
Collecting nvidia-curand-cu12==10.3.5.147 (from torch)
  Downloading nvidia_curand_cu12-10.3.5.147-py3-none-manylinux2014_x86_64.whl.metadat
Collecting nvidia-cusolver-cu12==11.6.1.9 (from torch)
  Downloading nvidia_cusolver_cu12-11.6.1.9-py3-none-manylinux2014_x86_64.whl.metadat
Collecting nvidia-cusparse-cu12==12.3.1.170 (from torch)
  Downloading nvidia_cusparse_cu12-12.3.1.170-py3-none-manylinux2014_x86_64.whl.metad
Requirement already satisfied: nvidia-cusparselt-cu12==0.6.2 in /usr/local/lib/python
Requirement already satisfied: nvidia-nccl-cu12==2.21.5 in /usr/local/lib/python3.11/
Requirement already satisfied: nvidia-nvtx-cu12==12.4.127 in /usr/local/lib/python3.1
Collecting nvidia-nvjitlink-cu12==12.4.127 (from torch)
  Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metada
Requirement already satisfied: triton==3.2.0 in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-p
Downloading bitsandbytes-0.45.4-py3-none-manylinux_2_24_x86_64.whl (76.0 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 76.0/76.0 MB 10.9 MB/s eta 0:00:00
Downloading nvidia_cublas_cu12-12.4.5.8-py3-none-manylinux2014_x86_64.whl (363.4 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 363.4/363.4 MB 3.8 MB/s eta 0:00:00
Downloading nvidia_cuda_cupti_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (13.8 M
```

```
!pip install datasets
```

```
Collecting datasets
  Downloading datasets-3.5.0-py3-none-any.whl.metadata (19 kB)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.11/dist-packages (f
Requirement already satisfied: pyarrow>=15.0.0 in /usr/local/lib/python3.11/dist-package
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 d
Requirement already satisfied: requests>=2.32.2 in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: tqdm>=4.66.3 in /usr/local/lib/python3.11/dist-packages (
Collecting xxhash (from datasets)
  Downloading xxhash-3.5.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.me
Collecting multiprocess<0.70.17 (from datasets)
  Downloading multiprocess-0.70.16-py311-none-any.whl.metadata (7.2 kB)
Collecting fsspec<=2024.12.0,>=2023.1.0 (from fsspec[http]<=2024.12.0,>=2023.1.0->datase
  Downloading fsspec-2024.12.0-py3-none-any.whl.metadata (11 kB)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.11/dist-packages (from
Requirement already satisfied: huggingface-hub>=0.24.0 in /usr/local/lib/python3.11/dist
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (fro
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dist-packages (f
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/python3.11/dist
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.11/d
```

```
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dis
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from
Downloading datasets-3.5.0-py3-none-any.whl (491 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 491.2/491.2 kB 19.8 MB/s eta 0:00:00
Downloading dill-0.3.8-py3-none-any.whl (116 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 116.3/116.3 kB 10.4 MB/s eta 0:00:00
Downloading fsspec-2024.12.0-py3-none-any.whl (183 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 183.9/183.9 kB 13.3 MB/s eta 0:00:00
Downloading multiprocess-0.70.16-py311-none-any.whl (143 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 143.5/143.5 kB 13.2 MB/s eta 0:00:00
Downloading xxhash-3.5.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (194
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 194.8/194.8 kB 16.9 MB/s eta 0:00:00
Installing collected packages: xxhash, fsspec, dill, multiprocess, 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 t
gcsfs 2025.3.2 requires fsspec==2025.3.2, but you have fsspec 2024.12.0 which is incompa
Successfully installed datasets-3.5.0 dill-0.3.8 fsspec-2024.12.0 multiprocess-0.70.16 x
```

```
import json

# Load preprocessed text file
with open("preprocessed_train.txt", "r", encoding="utf-8") as f:
    raw_data = f.read().split("=====\n")[:-1]  # Split based on your separator

# Extract buggy & fixed code pairs
dataset_dict = []
for sample in raw_data:
    buggy, fixed = sample.split("\nFIXED:\n")
    buggy = buggy.replace("BUGGY:\n", "").strip()
    fixed = fixed.strip()
    dataset_dict.append({"buggy_code": buggy, "fixed_code": fixed})

# Save to JSON
with open("preprocessed_train.json", "w", encoding="utf-8") as f:
    json.dump(dataset_dict, f, indent=4)

print("✅ Dataset converted to JSON!")
```

```
✅ Dataset converted to JSON!
```

```
from datasets import load_dataset
from transformers import AutoTokenizer

# Load dataset from JSON file
dataset = load_dataset("json", data_files="preprocessed_train.json")

# Load CodeT5 tokenizer
tokenizer = AutoTokenizer.from_pretrained("Salesforce/codet5-small")

def tokenize_function(examples):
    return tokenizer(
        examples["buggy_code"],
        text_target=examples["fixed_code"],  # For seq2seq models
        padding="max_length",
        truncation=True,
        max_length=512
    )

# Apply tokenization
tokenized_datasets = dataset.map(tokenize_function, batched=True)
tokenized_datasets = tokenized_datasets.remove_columns(["buggy_code", "fixed_code"])
```

```
Generating train split:        52364/0 [00:00<00:00, 134788.64 examples/s]
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public model
  warnings.warn(
tokenizer_config.json: 100%                                  1.48k/1.48k [00:00<00:00, 167kB/s]
vocab.json: 100%                                             703k/703k [00:00<00:00, 31.3MB/s]
merges.txt: 100%                                             294k/294k [00:00<00:00, 23.6MB/s]
added_tokens.json: 100%                                      2.00/2.00 [00:00<00:00, 194B/s]
special_tokens_map.json: 100%                                12.5k/12.5k [00:00<00:00, 1.10MB/s]
Map: 100%                                                    52364/52364 [00:43<00:00, 1312.59 examples/s]
```

```
from transformers import AutoModelForSeq2SeqLM, BitsAndBytesConfig
import torch
from peft import LoraConfig, get_peft_model
import bitsandbytes as bnb

# ✅ Configure 4-bit quantization
bnb_config = BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_quant_type="nf4",
```

```
    bnb_4bit_compute_dtype=torch.bfloat16,
    bnb_4bit_use_double_quant=True
)

# ✅ Load CodeT5 with 4-bit quantization
model = AutoModelForSeq2SeqLM.from_pretrained(
    "Salesforce/codet5-small",
    device_map="auto",
    quantization_config=bnb_config
)

# ✅ Apply LoRA to correct target modules
lora_config = LoraConfig(
    r=16,
    lora_alpha=32,
    target_modules=["q", "v", "k", "o", "wi", "wo"],  # ✅ Correct layers
    lora_dropout=0.05,
    bias="none",
    task_type="SEQ_2_SEQ_LM"
)

# ✅ Inject LoRA into CodeT5
model = get_peft_model(model, lora_config)
model.print_trainable_parameters()
```

```
config.json: 100%  1.57k/1.57k [00:00<00:00, 142kB/s]
pytorch_model.bin: 100%  242M/242M [00:01<00:00, 260MB/s]
model.safetensors: 100%  242M/242M [00:04<00:00, 21.7MB/s]
trainable params: 2,162,688 || all params: 62,654,976 || trainable%: 3.4517
```

```
def load_validation_data(buggy_path, fixed_path):
    with open(buggy_path, "r", encoding="utf-8") as f:
        buggy_code = f.readlines()

    with open(fixed_path, "r", encoding="utf-8") as f:
        fixed_code = f.readlines()

    return [{"input": b.strip(), "output": f.strip()} for b, f in zip(buggy_code, fixed_code

# Load validation data
validation_data = load_validation_data("/content/valid.buggy-fixed.buggy", "/content/valid.b
from transformers import AutoTokenizer

# Load CodeT5 tokenizer
tokenizer = AutoTokenizer.from_pretrained("Salesforce/codet5-small")

# Tokenization function
def preprocess_function(examples):
```

```
        model_inputs = tokenizer(examples["input"], padding="max_length", truncation=True, max_l
        labels = tokenizer(examples["output"], padding="max_length", truncation=True, max_length
        model_inputs["labels"] = labels["input_ids"]  # Assign tokenized outputs as labels
        return model_inputs

    # Apply tokenization
    tokenized_validation_data = [preprocess_function(example) for example in validation_data]
    from datasets import Dataset

    valid_dataset = Dataset.from_list(tokenized_validation_data)



    from transformers import TrainingArguments, Trainer

    training_args = TrainingArguments(
        output_dir="./qlora_codet5",
        evaluation_strategy="steps",
        save_strategy="epoch",
        per_device_train_batch_size=8,
        per_device_eval_batch_size=8,
        num_train_epochs=3,
        weight_decay=0.01,
        save_total_limit=2,
        logging_dir="./logs",
        report_to="none",
    )
    trainer = Trainer(
        model=model,
        args=training_args,
        train_dataset=tokenized_datasets["train"],
        eval_dataset=valid_dataset
    )

    trainer.train()
```

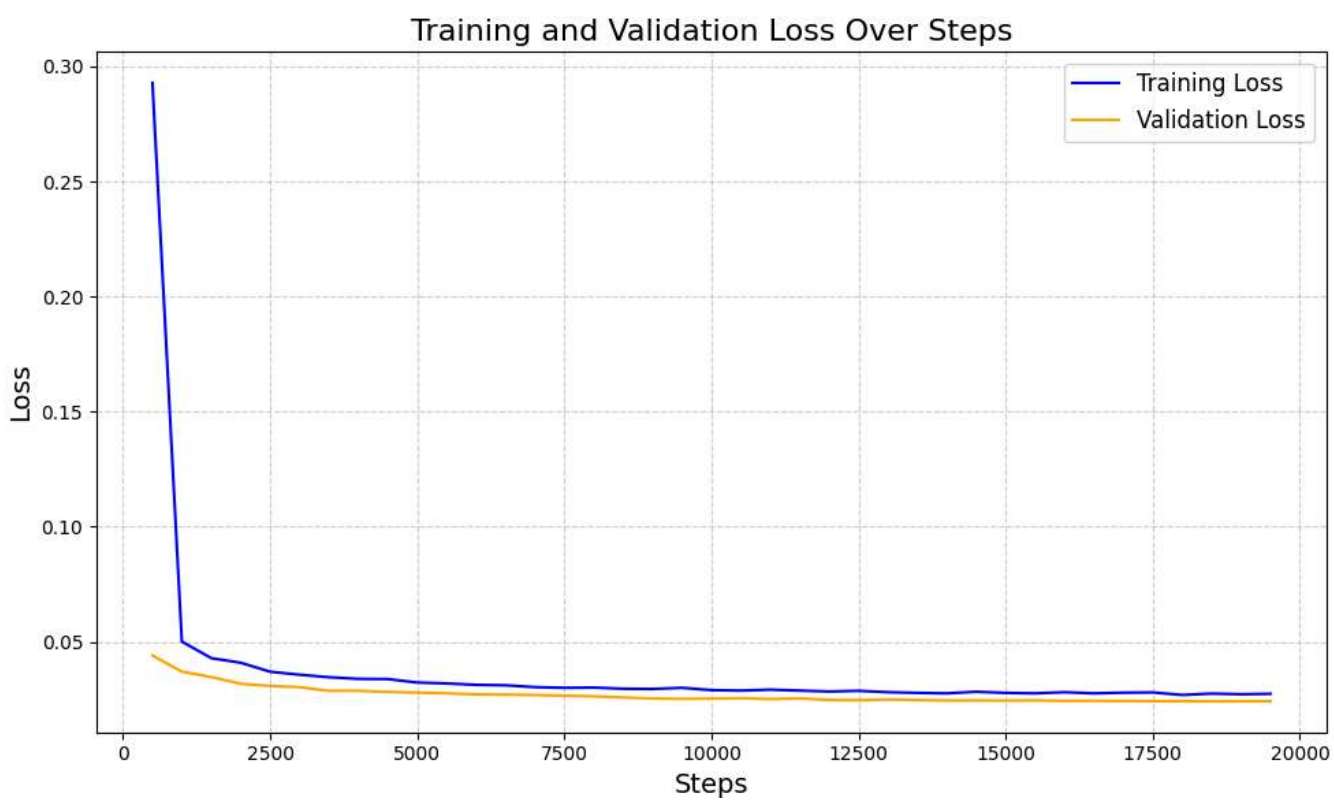
```
/usr/local/lib/python3.11/dist-packages/transformers/training_args.py:1611: FutureWar
  warnings.warn(
No label_names provided for model class `PeftModelForSeq2SeqLM`. Since `PeftModel` hi
Passing a tuple of `past_key_values` is deprecated and will be removed in Transformer
```

[19638/19638 3:32:14, Epoch 3/3]

Step	Training Loss	Validation Loss
500	0.292900	0.044040
1000	0.050100	0.037028
1500	0.042900	0.034695
2000	0.040900	0.031762
2500	0.037000	0.030819
3000	0.035700	0.030295
3500	0.034600	0.028678
4000	0.033900	0.028724
4500	0.033800	0.028215
5000	0.032300	0.027916
5500	0.031900	0.027606
6000	0.031300	0.027135
6500	0.031100	0.027035
7000	0.030300	0.026859
7500	0.030000	0.026580
8000	0.030100	0.026288
8500	0.029600	0.025809
9000	0.029500	0.025370
9500	0.030000	0.025184
10000	0.029000	0.025295
10500	0.028800	0.025502
11000	0.029200	0.025095
11500	0.028800	0.025403
12000	0.028400	0.024727
12500	0.028700	0.024606
13000	0.028100	0.024916
13500	0.027800	0.024698

```
14000          0.027600            0.024476
```

```
import matplotlib.pyplot as plt

# Data
steps = [500, 1000, 1500, 2000, 2500, 3000, 3500, 4000, 4500, 5000,
         5500, 6000, 6500, 7000, 7500, 8000, 8500, 9000, 9500, 10000,
         10500, 11000, 11500, 12000, 12500, 13000, 13500, 14000, 14500,
         15000, 15500, 16000, 16500, 17000, 17500, 18000, 18500, 19000, 19500]

training_loss = [0.292900, 0.050100, 0.042900, 0.040900, 0.037000, 0.035700,
                 0.034600, 0.033900, 0.033800, 0.032300, 0.031900, 0.031300,
                 0.031100, 0.030300, 0.030000, 0.030100, 0.029600, 0.029500,
                 0.030000, 0.029000, 0.028800, 0.029200, 0.028800, 0.028400,
                 0.028700, 0.028100, 0.027800, 0.027600, 0.028300, 0.027800,
                 0.027600, 0.028100, 0.027600, 0.027900, 0.028000, 0.026900,
                 0.027500, 0.027200, 0.027400]

validation_loss = [0.044040, 0.037028, 0.034695, 0.031762, 0.030819, 0.030295,
                   0.028678, 0.028724, 0.028215, 0.027916, 0.027606, 0.027135,
                   0.027035, 0.026859, 0.026580, 0.026288, 0.025809, 0.025370,
                   0.025184, 0.025295, 0.025502, 0.025095, 0.025403, 0.024727,
                   0.024606, 0.024916, 0.024698, 0.024476, 0.024511, 0.024435,
                   0.024514, 0.024310, 0.024340, 0.024270, 0.024190, 0.024190,
                   0.024114, 0.024135, 0.024191]

# Plotting
plt.figure(figsize=(10, 6))
plt.plot(steps, training_loss, label='Training Loss', color='blue')
plt.plot(steps, validation_loss, label='Validation Loss', color='orange')

# Adding labels and title
plt.title('Training and Validation Loss Over Steps', fontsize=16)
plt.xlabel('Steps', fontsize=14)
plt.ylabel('Loss', fontsize=14)
plt.legend(fontsize=12)

# Grid for better readability
plt.grid(True, linestyle='--', alpha=0.6)

# Show the plot
plt.tight_layout()
plt.show()
```




```
model.save_pretrained("codet5-lora-finetuned")
tokenizer.save_pretrained("codet5-lora-finetuned")
```

```
('codet5-lora-finetuned/tokenizer_config.json',
 'codet5-lora-finetuned/special_tokens_map.json',
 'codet5-lora-finetuned/vocab.json',
 'codet5-lora-finetuned/merges.txt',
 'codet5-lora-finetuned/added_tokens.json',
 'codet5-lora-finetuned/tokenizer.json')
```

```
metrics = trainer.evaluate()
print(metrics)
```

```
[819/819 02:11]
{'eval_loss': 0.02418339252471924, 'eval_runtime': 131.2444, 'eval_samples_per_second':
```

```
from transformers import pipeline

# Load the fine-tuned model
from peft import PeftModel
from transformers import AutoModelForSeq2SeqLM

base_model = AutoModelForSeq2SeqLM.from_pretrained("Salesforce/codet5-small")
model = PeftModel.from_pretrained(base_model, "codet5-lora-finetuned")
tokenizer = AutoTokenizer.from_pretrained("codet5-lora-finetuned")

# Define a pipeline for inference
pipe = pipeline("text2text-generation", model=model, tokenizer=tokenizer)

# Example input (as per your dataset)
input_text = "public int add(int a, int b) { return a + ; }"
output = pipe(input_text, max_length=64, do_sample=False)

print("Fixed Code:\n", output[0]['generated_text'])
```

```
Device set to use cuda:0
The model 'PeftModelForSeq2SeqLM' is not supported for text2text-generation. Supported m
Fixed Code:
 public int add ( int a , int b ) { return a + b ; }
```

```
from transformers import AutoTokenizer, AutoModelForSeq2SeqLM

tokenizer = AutoTokenizer.from_pretrained("Salesforce/codet5-small")
model = AutoModelForSeq2SeqLM.from_pretrained("Salesforce/codet5-small")
```

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public model
  warnings.warn(
tokenizer_config.json: 100%  1.48k/1.48k [00:00<00:00, 135kB/s]
vocab.json: 100%  703k/703k [00:00<00:00, 9.36MB/s]
merges.txt: 100%  294k/294k [00:00<00:00, 17.6MB/s]
added_tokens.json: 100%  2.00/2.00 [00:00<00:00, 237B/s]
special_tokens_map.json: 100%  12.5k/12.5k [00:00<00:00, 1.23MB/s]
config.json: 100%  1.57k/1.57k [00:00<00:00, 163kB/s]
pytorch_model.bin: 100%  242M/242M [00:02<00:00, 85.6MB/s]
```

```
from peft import PeftModel

# Assuming your LoRA adapter is in a folder called "your_adapter_folder"
model = PeftModel.from_pretrained(model, "/content/codet5-lora-finetuned")
```

```
/usr/local/lib/python3.11/dist-packages/peft/peft_model.py:599: UserWarning: Found missi
  warnings.warn(f"Found missing adapter keys while loading the checkpoint: {missing_keys
```

```
def generate_fix(buggy_code):
    inputs = tokenizer(buggy_code, return_tensors="pt", padding=True)
    output = model.generate(**inputs, max_length=512)
    fixed_code = tokenizer.decode(output[0], skip_special_tokens=True)
    return fixed_code

buggy_code = """public static TYPE_1 init ( java.lang.String name , java.util.Date date ) {
fixed_code = generate_fix(buggy_code)
print(fixed_code)
```

```
public static TYPE_1 init ( java.lang.String name , java.util.Date date ) { TYPE_1 VAR_1
```

```
import torch

# Load tokenizer and model from your saved directory
from transformers import AutoTokenizer, AutoModelForSeq2SeqLM
from peft import PeftModel

# Set device
device = "cuda" if torch.cuda.is_available() else "cpu"

# Path where you saved your model and tokenizer
model_path = "/content/codet5-lora-finetuned"

# Load tokenizer
tokenizer = AutoTokenizer.from_pretrained(model_path)

# Load base model
model = AutoModelForSeq2SeqLM.from_pretrained(model_path)
model = PeftModel.from_pretrained(model, model_path)

# Send model to device
model = model.to(device)
model.eval()

# Generation function
def generate_fix(buggy_code):
    inputs = tokenizer(buggy_code, return_tensors="pt", padding=True, truncation=True).to(de
```

```
    with torch.no_grad():
        outputs = model.generate(
            **inputs,
            max_new_tokens=128,
            num_beams=5,
            early_stopping=True
        )

    # Decode the output
    fixed_code = tokenizer.decode(outputs[0], skip_special_tokens=True)
    return fixed_code

# Example usage:
buggy = """public static TYPE_1 init ( java.lang.String name , java.util.Date date ) { TYPE_

fixed = generate_fix(buggy)
print(f"Fixed Code:\n{fixed}")
```

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public model
  warnings.warn(
Fixed Code:
public static TYPE_1 init ( java.lang.String name , java.util.Date date ) { TYPE_1 VAR_1
```

```
buggy_samples = [
    "public static TYPE_1 init ( java.lang.String name , java.util.Date date ) { TYPE_1 VAR_
    "public static int add(int a, int b) { return a - b; }",
    # Add more buggy code samples here
]

for buggy_code in buggy_samples:
    fixed_code = generate_fix(buggy_code)
    print(f"Buggy Code:\n{buggy_code}\n")
    print(f"Fixed Code:\n{fixed_code}\n")
    print("="*50)
```

```
Buggy Code:
public static TYPE_1 init ( java.lang.String name , java.util.Date date ) { TYPE_1 VAR_1

Fixed Code:
public static TYPE_1 init ( java.lang.String name , java.util.Date date ) { TYPE_1 VAR_1

==================================================
Buggy Code:
public static int add(int a, int b) { return a - b; }
```

```
Fixed Code:
public static int add ( int a , int b ) { return a + b ; }

==================================================
```

```
pip install nltk
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.11/dist-packages (3.9.1)
Requirement already satisfied: click in /usr/local/lib/python3.11/dist-packages (from nl
Requirement already satisfied: joblib in /usr/local/lib/python3.11/dist-packages (from r
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from nlt
```

```
import nltk
from nltk.translate.bleu_score import sentence_bleu
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
True
```

```
def compute_bleu(reference, prediction):
    """
    Compute BLEU score between a reference and a prediction.
    """
    reference_tokens = nltk.word_tokenize(reference)
    prediction_tokens = nltk.word_tokenize(prediction)
    score = sentence_bleu([reference_tokens], prediction_tokens)
    return score
```

```
# Read buggy inputs
with open("/content/test.buggy-fixed.buggy", "r", encoding="utf-8") as f:
    buggy_codes = [line.strip() for line in f if line.strip()]

# Read ground truth fixed codes
with open("/content/test.buggy-fixed.fixed", "r", encoding="utf-8") as f:
    ground_truth_fixed_codes = [line.strip() for line in f if line.strip()]


predictions = []

for buggy_code in buggy_codes:
    inputs = tokenizer(buggy_code, return_tensors="pt", padding=True, truncation=True).to(mo
    with torch.no_grad():
        outputs = model.generate(**inputs, max_new_tokens=256)
```

```
    pred_code = tokenizer.decode(outputs[0], skip_special_tokens=True)
    predictions.append(pred_code)
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-7-5b759f3bbed5> in <cell line: 0>()
     13     inputs = tokenizer(buggy_code, return_tensors="pt", padding=True,
truncation=True).to(model.device)
     14     with torch.no_grad():
---> 15         outputs = model.generate(**inputs, max_new_tokens=256)
     16     pred_code = tokenizer.decode(outputs[0], skip_special_tokens=True)
     17     predictions.append(pred_code)

                                 24 frames
/usr/local/lib/python3.11/dist-packages/torch/nn/modules/linear.py in forward(self,
input)
    123
    124     def forward(self, input: Tensor) -> Tensor:
    126
    127     def extra_repr(self) -> str:

KeyboardInterrupt:
```

```
len(predictions)
```

```
328
```

```
ground_truth_fixed_codes = ground_truth_fixed_codes[:328]

len(ground_truth_fixed_codes)
```

```
328
```

```
from nltk.translate.bleu_score import corpus_bleu

# Tokenize the references and hypotheses
references = [[ref.split()] for ref in ground_truth_fixed_codes]  # List of list of referenc
hypotheses = [pred.split() for pred in predictions]

# Calculate BLEU
bleu_score = corpus_bleu(references, hypotheses)

print(f"BLEU Score: {bleu_score * 100:.2f}")
```

```
BLEU Score: 91.03
```

```
import seaborn as sns

sns.violinplot(bleu_score)
plt.title('Distribution of BLEU Scores')
plt.show()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-23-1ef957b04077> in <cell line: 0>()
      1 import seaborn as sns
      2
----> 3 sns.violinplot(bleu_score)
      4 plt.title('Distribution of BLEU Scores')
      5 plt.show()

                                 4 frames
/usr/local/lib/python3.11/dist-packages/seaborn/_base.py in
_assign_variables_wideform(self, data, **kwargs)
    724
    725         # Determine if the data object actually has any data in it
--> 726         empty = data is None or not len(data)
    727
    728         # Then, determine if we have "flat" data (a single vector)

TypeError: object of type 'float' has no len()
```

+ Code + Text

```
from huggingface_hub import notebook_login

notebook_login()
```

```
from peft import PeftModel

repo_name = "qlora-codet5-java-bugfix"  # 🔥 you can change this name as you want


model.push_to_hub(repo_name)
tokenizer.push_to_hub(repo_name)
```

```
adapter_model.safetensors: 100%                 8.68M/8.68M [00:00<00:00, 30.2MB/s]
```