```
import pandas as pd

def prepare_detection_dataset(buggy_file, fixed_file, output_csv):
    with open(buggy_file, 'r', encoding='utf-8') as bf, open(fixed_file, 'r', encoding='utf-8') as ff:
        buggy_codes = [line.strip() for line in bf]
        fixed_codes = [line.strip() for line in ff]

    # Create a dataset
    data = []
    for buggy in buggy_codes:
        data.append({'code': buggy, 'label': 1})  # Buggy: 1
    for fixed in fixed_codes:
        data.append({'code': fixed, 'label': 0})  # Fixed: 0

    # Save to CSV
    df = pd.DataFrame(data)
    df.to_csv(output_csv, index=False)
    print(f"✅ Detection dataset created and saved at: {output_csv}")

# Example usage:
prepare_detection_dataset(
    buggy_file='/content/train.buggy-fixed.buggy',
    fixed_file='/content/train.buggy-fixed.fixed',
    output_csv='/content/detection_dataset.csv'
)
```

```
✅ Detection dataset created and saved at: /content/detection_dataset.csv
```

```
import pandas as pd

def prepare_detection_dataset(buggy_file, fixed_file, output_csv):
    with open(buggy_file, 'r', encoding='utf-8') as bf, open(fixed_file, 'r', encoding='utf-8') as ff:
        buggy_codes = [line.strip() for line in bf]
        fixed_codes = [line.strip() for line in ff]

    # Create a dataset
    data = []
    for buggy in buggy_codes:
        data.append({'code': buggy, 'label': 1})  # Buggy: 1
    for fixed in fixed_codes:
        data.append({'code': fixed, 'label': 0})  # Fixed: 0

    # Save to CSV
    df = pd.DataFrame(data)
    df.to_csv(output_csv, index=False)
    print(f"✅ Detection dataset created and saved at: {output_csv}")

# Example usage:
prepare_detection_dataset(
    buggy_file="/content/valid.buggy-fixed.buggy",
    fixed_file="/content/valid.buggy-fixed.fixed",
    output_csv='/content/detection_datasetval.csv'
)
```

```
✅ Detection dataset created and saved at: /content/detection_datasetval.csv
```

```
!pip install datasets
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (from datasets) (2.2.2)
Requirement already satisfied: requests>=2.32.2 in /usr/local/lib/python3.11/dist-packages (from datasets) (2.32.3)
Requirement already satisfied: tqdm>=4.66.3 in /usr/local/lib/python3.11/dist-packages (from datasets) (4.67.1)
Collecting xxhash (from datasets)
  Downloading xxhash-3.5.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (12 kB)
Collecting multiprocess<0.70.17 (from datasets)
  Downloading multiprocess-0.70.16-py311-none-any.whl.metadata (7.2 kB)
Collecting fsspec<=2024.12.0,>=2023.1.0 (from fsspec[http]<=2024.12.0,>=2023.1.0->datasets)
  Downloading fsspec-2024.12.0-py3-none-any.whl.metadata (11 kB)
```

```
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (1.5.0)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (6.3.1)
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (0.3.1)
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (1.18.3)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.24.0->d
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests>=2.32.2->datasets)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests>=2.32.2->datasets) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests>=2.32.2->datasets) (2.3.0
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests>=2.32.2->datasets) (2025.
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas->datasets) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas->datasets) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas->datasets) (2025.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas->datasets) (1
Downloading datasets-3.5.0-py3-none-any.whl (491 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 491.2/491.2 kB 27.8 MB/s eta 0:00:00
Downloading dill-0.3.8-py3-none-any.whl (116 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 116.3/116.3 kB 10.9 MB/s eta 0:00:00
Downloading fsspec-2024.12.0-py3-none-any.whl (183 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 183.9/183.9 kB 16.2 MB/s eta 0:00:00
Downloading multiprocess-0.70.16-py311-none-any.whl (143 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 143.5/143.5 kB 13.4 MB/s eta 0:00:00
Downloading xxhash-3.5.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (194 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 194.8/194.8 kB 16.7 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 that are installed. This behaviour is the sour
torch 2.6.0+cu124 requires nvidia-cublas-cu12==12.4.5.8; platform_system == "Linux" and platform_machine == "x86_64", but you have nv
torch 2.6.0+cu124 requires nvidia-cuda-cupti-cu12==12.4.127; platform_system == "Linux" and platform_machine == "x86_64", but you hav
torch 2.6.0+cu124 requires nvidia-cuda-nvrtc-cu12==12.4.127; platform_system == "Linux" and platform_machine == "x86_64", but you hav
torch 2.6.0+cu124 requires nvidia-cuda-runtime-cu12==12.4.127; platform_system == "Linux" and platform_machine == "x86_64", but you h
torch 2.6.0+cu124 requires nvidia-cudnn-cu12==9.1.0.70; platform_system == "Linux" and platform_machine == "x86_64", but you have nvi
torch 2.6.0+cu124 requires nvidia-cufft-cu12==11.2.1.3; platform_system == "Linux" and platform_machine == "x86_64", but you have nvi
torch 2.6.0+cu124 requires nvidia-curand-cu12==10.3.5.147; platform_system == "Linux" and platform_machine == "x86_64", but you have
torch 2.6.0+cu124 requires nvidia-cusolver-cu12==11.6.1.9; platform_system == "Linux" and platform_machine == "x86_64", but you have
torch 2.6.0+cu124 requires nvidia-cusparse-cu12==12.3.1.170; platform_system == "Linux" and platform_machine == "x86_64", but you hav
torch 2.6.0+cu124 requires nvidia-nvjitlink-cu12==12.4.127; platform_system == "Linux" and platform_machine == "x86_64", but you have
gcsfs 2025.3.2 requires fsspec==2025.3.2, but you have fsspec 2024.12.0 which is incompatible.
Successfully installed datasets-3.5.0 dill-0.3.8 fsspec-2024.12.0 multiprocess-0.70.16 xxhash-3.5.0
```

```
from transformers import AutoTokenizer
from datasets import load_dataset

# Load CodeT5 tokenizer
tokenizer = AutoTokenizer.from_pretrained("Salesforce/codet5-small")  # or use your fine-tuned one if needed

# Load your CSV as Huggingface Dataset
dataset = load_dataset("csv", data_files="detection_dataset.csv")

def tokenize_function(examples):
    return tokenizer(
        examples["code"],  # 'code' is the column with code snippets
        padding="max_length",
        truncation=True,
        max_length=512,
    )

# Tokenize
tokenized_dataset = dataset.map(tokenize_function, batched=True)
```

```
/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://huggingface.co/settings/tokens), set it as secre
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 models or datasets.
  warnings.warn(
```

```
tokenizer_config.json: 100%  1.48k/1.48k [00:00<00:00, 164kB/s]
vocab.json: 100%  703k/703k [00:00<00:00, 26.1MB/s]
merges.txt: 100%  294k/294k [00:00<00:00, 30.1MB/s]
added_tokens.json: 100%  2.00/2.00 [00:00<00:00, 138B/s]
special_tokens_map.json: 100%  12.5k/12.5k [00:00<00:00, 1.05MB/s]
Map: 100%  104728/104728 [00:50<00:00, 2567.93 examples/s]
```

```
tokenized_dataset
```

```
DatasetDict({
    train: Dataset({
        features: ['code', 'label', 'input_ids', 'attention_mask'],
        num_rows: 104728
    })
})
```

```
tokenized_dataset_val
```

```
DatasetDict({
    train: Dataset({
        features: ['code', 'label', 'input_ids', 'attention_mask'],
        num_rows: 13092
    })
})
```

```
# Load your CSV as Huggingface Dataset
dataset = load_dataset("csv", data_files="/content/detection_datasetval.csv")

def tokenize_function(examples):
    return tokenizer(
        examples["code"],  # 'code' is the column with code snippets
        padding="max_length",
        truncation=True,
        max_length=512,
    )

# Tokenize
tokenized_dataset_val = dataset.map(tokenize_function, batched=True)
```

```
Generating train split:  13092/0 [00:00<00:00, 93158.93 examples/s]
Map: 100%  13092/13092 [00:09<00:00, 1173.93 examples/s]
```

```
from transformers import AutoModelForSequenceClassification

# Load CodeT5 model for classification
model = AutoModelForSequenceClassification.from_pretrained(
    "Salesforce/codet5-small",
    num_labels=2  # 2 classes: buggy (1) and fixed (0)
)
```

```
config.json: 100%  1.57k/1.57k [00:00<00:00, 146kB/s]
pytorch_model.bin: 100%  242M/242M [00:01<00:00, 166MB/s]
Some weights of T5ForSequenceClassification were not initialized from the model checkpoint at Salesforce/codet5-small and are newly init
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
```

```
from peft import LoraConfig, get_peft_model, prepare_model_for_kbit_training

# QLoRA config
lora_config = LoraConfig(
```

```
    r=16,
    lora_alpha=32,
    target_modules=["q", "v"],  # if you want to specify; or leave default
    lora_dropout=0.05,
    bias="none",
    task_type="SEQ_CLS"  # Sequence Classification
)

# Prepare model for k-bit training
model = prepare_model_for_kbit_training(model)

# Apply LoRA
model = get_peft_model(model, lora_config)

# Check trainable params
model.print_trainable_parameters()
```

```
trainable params: 589,824 || all params: 61,345,794 || trainable%: 0.9615
```

```
from transformers import Trainer, TrainingArguments



training_args = TrainingArguments(
    output_dir="./results",
    evaluation_strategy="epoch",   # evaluate every epoch
    save_strategy="epoch",          # save checkpoint every epoch
    learning_rate=2e-4,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=16,
    num_train_epochs=1,
    weight_decay=0.01,
    logging_dir="./logs",
    logging_steps=50,
    save_total_limit=2,             # keep only last 2 checkpoints
    load_best_model_at_end=True,
    metric_for_best_model="accuracy",
    greater_is_better=True,
    report_to="none"                # (disable wandb/huggingface logging if you don't use them)
)


trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_dataset["train"],
    eval_dataset=tokenized_dataset_val["train"],  # or separate val split if you have
    tokenizer=tokenizer,
)

# Train 🚀
trainer.train()
```

```
/usr/local/lib/python3.11/dist-packages/transformers/training_args.py:1611: FutureWarning: `evaluation_strategy` is deprecated and will
  warnings.warn(
<ipython-input-15-00a7d55d9a89>:24: FutureWarning: `tokenizer` is deprecated and will be removed in version 5.0.0 for `Trainer.__init__`
  trainer = Trainer(
No label_names provided for model class `PeftModelForSequenceClassification`. Since `PeftModel` hides base models input arguments, if la
```

[6546/6546 1:29:54, Epoch 1/1]

Epoch	Training Loss	Validation Loss
1	0.699800	0.691609

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/usr/local/lib/python3.11/dist-packages/transformers/trainer.py in _determine_best_metric(self, metrics, trial)
   3161             try:
-> 3162                 metric_value = metrics[metric_to_check]
   3163             except KeyError as exc:

KeyError: 'eval_accuracy'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
                                          4 frames
/usr/local/lib/python3.11/dist-packages/transformers/trainer.py in _determine_best_metric(self, metrics, trial)
   3162                 metric_value = metrics[metric_to_check]
   3163             except KeyError as exc:
-> 3164                 raise KeyError(
   3165                     f"The `metric_for_best_model` training argument is set to '{metric_to_check}', which is not found in the
evaluation metrics. "
   3166                     f"The available evaluation metrics are: {list(metrics.keys())}. Consider changing the
`metric_for_best_model` via the TrainingArguments."

KeyError: "The `metric_for_best_model` training argument is set to 'eval_accuracy', which is not found in the evaluation metrics. The
available evaluation metrics are: ['eval_loss']. Consider changing the `metric_for_best_model` via the TrainingArguments."
```

Next steps: Explain error

```
# Save the model and tokenizer
trainer.save_model("./saved_codet5_detection_model")
tokenizer.save_pretrained("./saved_codet5_detection_model")
```

```
('./saved_codet5_detection_model/tokenizer_config.json',
 './saved_codet5_detection_model/special_tokens_map.json',
 './saved_codet5_detection_model/vocab.json',
 './saved_codet5_detection_model/merges.txt',
 './saved_codet5_detection_model/added_tokens.json',
 './saved_codet5_detection_model/tokenizer.json')
```

```
from huggingface_hub import notebook_login

notebook_login()  # Logs you in
```

```
repo_name = "qlora-codet5-java-bugdetection"
```

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

```
adapter_model.safetensors: 100%  2.37M/2.37M [00:01<00:00, 8.21MB/s]
README.md: 100%  5.17k/5.17k [00:00<00:00, 514kB/s]
CommitInfo(commit_url='https://huggingface.co/nairabhiram907/qlora-codet5-java-
bugdetection/commit/af9ee7a1951ca7caef1be7b85e401de409c3dbf7', commit_message='Upload tokenizer', commit_description='',
oid='af9ee7a1951ca7caef1be7b85e401de409c3dbf7', pr_url=None, repo_url=RepoUrl('https://huggingface.co/nairabhiram907/qlora-codet5-java-
```

```
from transformers import AutoTokenizer, AutoModelForSequenceClassification
import torch

# Load the saved model
```

```
model_path = "/content/saved_codet5_detection_model"
tokenizer = AutoTokenizer.from_pretrained(model_path)
model = AutoModelForSequenceClassification.from_pretrained(model_path)
model.eval()
```

```
      (dropout): Dropout(p=0.1, inplace=False)
      (act): ReLU()
    )
    (layer_norm): T5LayerNorm()
    (dropout): Dropout(p=0.1, inplace=False)
  )
 )
)
(1-5): 5 x T5Block(
 (layer): ModuleList(
  (0): T5LayerSelfAttention(
   (SelfAttention): T5Attention(
    (q): lora.Linear(
     (base_layer): Linear(in_features=512, out_features=512, bias=False)
     (lora_dropout): ModuleDict(
      (default): Dropout(p=0.05, inplace=False)
     )
     (lora_A): ModuleDict(
      (default): Linear(in_features=512, out_features=16, bias=False)
     )
     (lora_B): ModuleDict(
      (default): Linear(in_features=16, out_features=512, bias=False)
     )
     (lora_embedding_A): ParameterDict()
     (lora_embedding_B): ParameterDict()
     (lora_magnitude_vector): ModuleDict()
    )
    (k): Linear(in_features=512, out_features=512, bias=False)
    (v): lora.Linear(
     (base_layer): Linear(in_features=512, out_features=512, bias=False)
     (lora_dropout): ModuleDict(
      (default): Dropout(p=0.05, inplace=False)
     )
     (lora_A): ModuleDict(
      (default): Linear(in_features=512, out_features=16, bias=False)
     )
     (lora_B): ModuleDict(
      (default): Linear(in_features=16, out_features=512, bias=False)
     )
     (lora_embedding_A): ParameterDict()
     (lora_embedding_B): ParameterDict()
     (lora_magnitude_vector): ModuleDict()
    )
    (o): Linear(in_features=512, out_features=512, bias=False)
   )
   (layer_norm): T5LayerNorm()
   (dropout): Dropout(p=0.1, inplace=False)
  )
  (1): T5LayerCrossAttention(
   (EncDecAttention): T5Attention(
    (q): lora.Linear(
     (base_layer): Linear(in_features=512, out_features=512, bias=False)
     (lora_dropout): ModuleDict(
      (default): Dropout(p=0.05, inplace=False)
     )
     (lora_A): ModuleDict(
      (default): Linear(in_features=512, out_features=16, bias=False)
     )
```

```
def detect_bug(code_snippet):
    inputs = tokenizer(code_snippet, return_tensors="pt", truncation=True, padding=True)
    with torch.no_grad():
        outputs = model(**inputs)
        probs = torch.nn.functional.softmax(outputs.logits, dim=-1)
        prediction = torch.argmax(probs, dim=1).item()
```