```
#!pip install transformers datasets pandas
from google.colab import files
uploaded = files.upload()
```

```
Choose files No file chosen  Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving radiology_5000_samples.csv to radiology_5000_samples.csv
```

```
import os
print(os.listdir())
```

```
['.config', 'radiology_5000_samples.csv', 'sample_data']
```

```
from transformers import AutoTokenizer, AutoModelForSeq2SeqLM

model_name = "t5-small"

tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForSeq2SeqLM.from_pretrained(model_name)
```

```
Loading weights: 100% 131/131 [00:00<00:00, 530.27it/s, Materializing param=shared.weight]
```

```
from transformers import Trainer, TrainingArguments, DataCollatorForSeq2Seq, AutoTokenizer, AutoModelForSeq2SeqLM
from datasets import load_dataset, Dataset
import random
import pandas as pd
import os # Import os for file path checks

# Re-initialize tokenizer and model in case runtime state was lost
model_name = "t5-small"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForSeq2SeqLM.from_pretrained(model_name)

# Determine the correct CSV file name
```

```
[ ]
file_name = "radiology_5000.csv"
if not os.path.exists(file_name):
    print(f"Warning: {file_name} not found. Checking other possible names in {os.getcwd()}.")
    possible_files = [f for f in os.listdir('.') if 'radiology_5000' in f and f.endswith('.csv')]
    if possible_files:
        # Prioritize exact match, then samples, then other variations
        if 'radiology_5000.csv' in possible_files:
            file_name = 'radiology_5000.csv'
        elif 'radiology_5000_samples.csv' in possible_files:
            file_name = 'radiology_5000_samples.csv'
        elif 'radiology_5000_samples (1).csv' in possible_files:
            file_name = 'radiology_5000_samples (1).csv'
        elif 'radiology_5000_samples (1) (1).csv' in possible_files:
            file_name = 'radiology_5000_samples (1) (1).csv'
        else:
            file_name = possible_files[0] # Fallback to first found if others not matched
        print(f"Using {file_name} instead.")
    else:
        raise FileNotFoundError(f"No file resembling 'radiology_5000.csv' found in {os.getcwd()}. Please upload it.")

# Load CSV using pandas and convert to datasets.Dataset
df = pd.read_csv(file_name)
dataset = Dataset.from_pandas(df)

train_test = dataset.train_test_split(test_size=0.1)
train_set = train_test["train"]
test_set = train_test["test"]

# --- Preprocessing function definition
def dual_preprocess(examples):
    inputs = []
    targets = []
    for i in range(len(examples["findings"])):
        f = examples["findings"][i]
        imp = examples["impression"][i]
        if random.random() < 0.5:
            inputs.append("summarize medically: " + f)
            targets.append(imp)
        else:
```

```
                inputs.append("summarize simply: " + f)
                targets.append("This means: " + imp)
        model_inputs = tokenizer(
            inputs,
            max_length=256,
            truncation=True,
            padding="max_length"
        )
        labels = tokenizer(
            targets,
            max_length=64,
            truncation=True,
            padding="max_length"
        )
        model_inputs["labels"] = labels["input_ids"]
        return model_inputs

    # Tokenization
    train_tokenized = train_set.map(dual_preprocess, batched=True)
    test_tokenized = test_set.map(dual_preprocess, batched=True)


    training_args = TrainingArguments(
        output_dir="./results",
        learning_rate=3e-5,
        per_device_train_batch_size=4,
        per_device_eval_batch_size=4,
        num_train_epochs=4,
        weight_decay=0.01,
        logging_steps=50,
        save_strategy="no",
        report_to="none"
    )

    data_collator = DataCollatorForSeq2Seq(tokenizer, model=model)

    trainer = Trainer(
        model=model,
        args=training_args,
```

```
    args=training_args,
    train_dataset=train_tokenized,
    eval_dataset=test_tokenized,
    data_collator=data_collator
)

trainer.train()
```

```
Loading weights: 100% 131/131 [00:00<00:00, 481.85it/s, Materializing param=shared.weight]
Warning: radiology_5000.csv not found. Checking other possible names in /content.
Using radiology_5000_samples.csv instead.
Map: 100% 4500/4500 [00:00<00:00, 5034.88 examples/s]
Map: 100% 500/500 [00:00<00:00, 4299.00 examples/s]
[4500/4500 07:56, Epoch 4/4]
```

| Step | Training Loss |
|---|---|
| 50 | 7.019860 |
| 100 | 1.150077 |
| 150 | 0.602765 |
| 200 | 0.429446 |
| 250 | 0.276302 |
| 300 | 0.164054 |
| 350 | 0.105594 |
| 400 | 0.072693 |
| 450 | 0.050868 |
| 500 | 0.042152 |
| 550 | 0.030666 |
| 600 | 0.023577 |
| 650 | 0.017357 |

| | |
|---|---|
| 900 | 0.006986 |
| 950 | 0.007097 |
| 1000 | 0.005968 |
| 1050 | 0.005663 |
| 1100 | 0.003900 |
| 1150 | 0.004187 |
| 1200 | 0.004010 |
| 1250 | 0.003752 |
| 1300 | 0.003670 |
| 1350 | 0.003278 |
| 1400 | 0.002969 |
| 1450 | 0.003392 |
| 1500 | 0.002690 |
| 1550 | 0.002169 |
| 1600 | 0.001640 |
| 1650 | 0.001894 |
| 1700 | 0.002476 |
| 1750 | 0.001688 |
| 1800 | 0.002154 |
| 1850 | 0.001916 |
| 1900 | 0.001888 |
| 1950 | 0.001297 |
| 2000 | 0.001591 |

| | |
|---|---|
| 3450 | 0.000683 |
| 3500 | 0.000733 |
| 3550 | 0.001331 |
| 3600 | 0.000539 |
| 3650 | 0.000564 |
| 3700 | 0.000668 |
| 3750 | 0.000704 |
| 3800 | 0.000594 |
| 3850 | 0.000480 |
| 3900 | 0.000687 |
| 3950 | 0.000695 |
| 4000 | 0.000801 |
| 4050 | 0.000642 |
| 4100 | 0.000466 |
| 4150 | 0.000652 |
| 4200 | 0.000631 |
| 4250 | 0.000524 |
| 4300 | 0.000718 |
| 4350 | 0.000557 |
| 4400 | 0.000543 |
| 4450 | 0.000467 |
| 4500 | 0.000457 |

```
TrainOutput(global_step=4500, training_loss=0.11279031479193105, metrics={'train_runtime': 477.7472, 'train_samples_per_second': 37.677, 'train_steps_per_second
```

```
4500          0.000457
```

```
TrainOutput(global_step=4500, training_loss=0.11279031479193105, metrics={'train_runtime': 477.7472, 'train_samples_per_second': 37.677, 'train_steps_per_second': 9.419,
'total_flos': 1218076213248000.0, 'train_loss': 0.11279031479193105, 'epoch': 4.0})
```

```
print(len(train_tokenized))
```

```
4500
```

Double-click (or enter) to edit

```
import torch

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model.to(device)

def summarize(text):
  inputs = tokenizer(
      text,
      return_tensors="pt",
      truncation=True,
      max_length=256
  ).to(device)

  outputs = model.generate(**inputs, max_length=50)

  return tokenizer.decode(outputs[0], skip_special_tokens=True)
```

```
print(summarize("Patchy opacity in left lower lobe. Possible pneumonia."))
print(summarize("Cardiac silhouette enlarged. Pulmonary edema present."))
print(summarize("No acute cardiopulmonary abnormality."))
```

```
test_cases = [
    "Large consolidation in left lower lung. Suspicious for pneumonia."
```

```
[ ]
test_cases = [
        "Large consolidation in left lower lung. Suspicious for pneumonia.",
        "Cardiac silhouette enlarged. Pulmonary edema present.",
        "Small nodule in right upper lobe.",
        "Bilateral infiltrates. Possible viral pneumonia."
                ]

for t in test_cases:
  print("Input:", t)
  print("Summary:", summarize(t))
  print()
```

```
Input: Large consolidation in left lower lung. Suspicious for pneumonia.
Summary: pneumonia.

Input: Cardiac silhouette enlarged. Pulmonary edema present.
Summary: . Cardiac silhouette enlarged.

Input: Small nodule in right upper lobe.
Summary: Small nodule in right upper lobe.

Input: Bilateral infiltrates. Possible viral pneumonia.
Summary: Viral pneumonia suspected.
```

```
[ ]
import matplotlib.pyplot as plt

losses = [log["loss"] for log in trainer.state.log_history if "loss" in log]

plt.plot(losses)
plt.xlabel("Training Steps")
plt.ylabel("Loss")
plt.title("Training Loss Curve")
plt.show()
```

```
[ ]
def doctor_summary(text):
```

```
[ ]
def doctor_summary(text):
  inputs = tokenizer("summarize medically: " + text,
                     return_tensors="pt",
                     truncation=True,
                     max_length=256).to(model.device)

  outputs = model.generate(**inputs, max_length=50)
  return tokenizer.decode(outputs[0], skip_special_tokens=True)


def patient_summary(text):
  inputs = tokenizer("summarize simply: " + text,
                     return_tensors="pt",
                     truncation=True,
                     max_length=256).to(model.device)

  outputs = model.generate(**inputs, max_length=50)
  return tokenizer.decode(outputs[0], skip_special_tokens=True)
```

```
[ ]
text = "Patchy opacity in left lower lobe. Possible pneumonia."

print("Doctor Summary:", doctor_summary(text))
print("Patient Summary:", patient_summary(text))
```

```
Doctor Summary: Possible pneumonia.
Patient Summary: This means: Left lower lobe pneumonia.
```