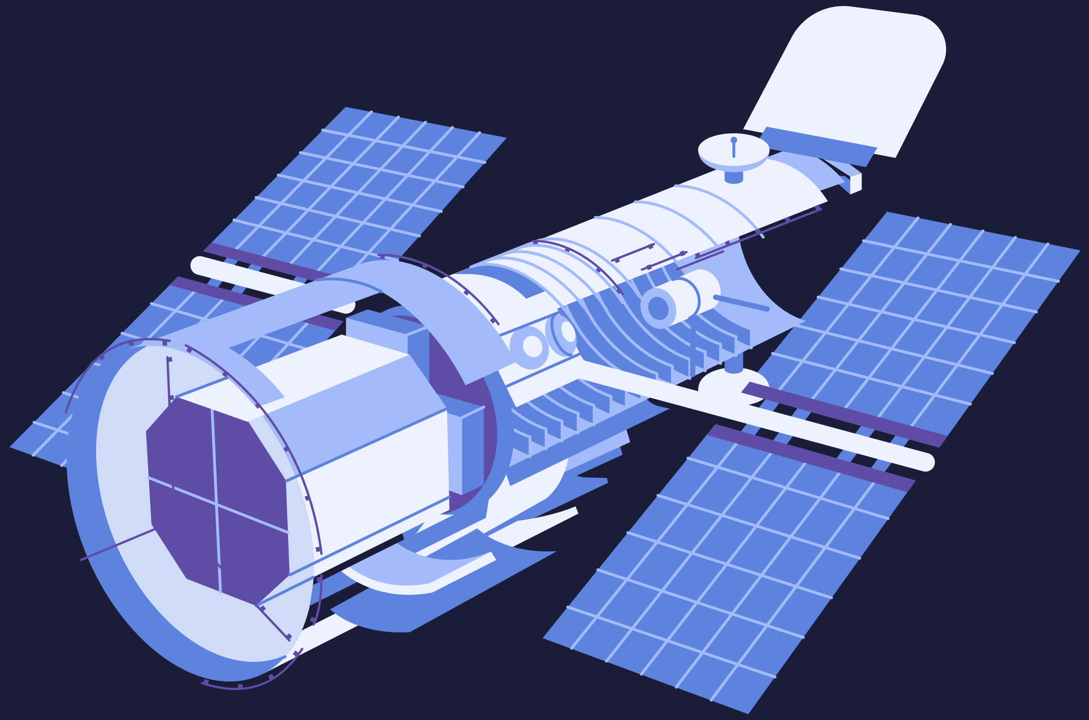
Group 2

FINE-TUNING A MODEL FOR SUMMARIZATION TASK

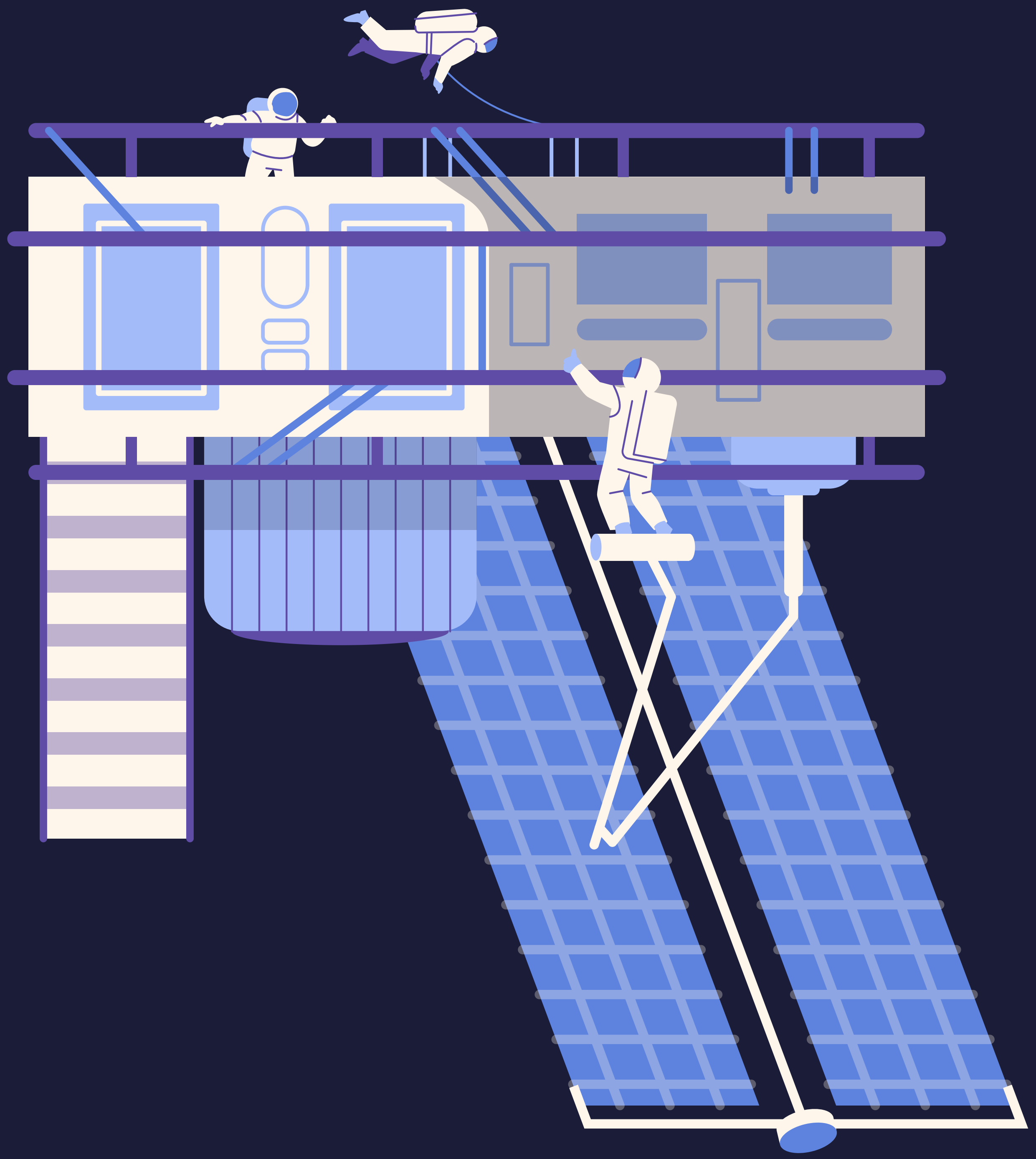
Mashael Aljuhani
Noura Aldawsari
Ebtsam Asiri
Shahad Alhmoud
Nouf Alomari

CONTENT:



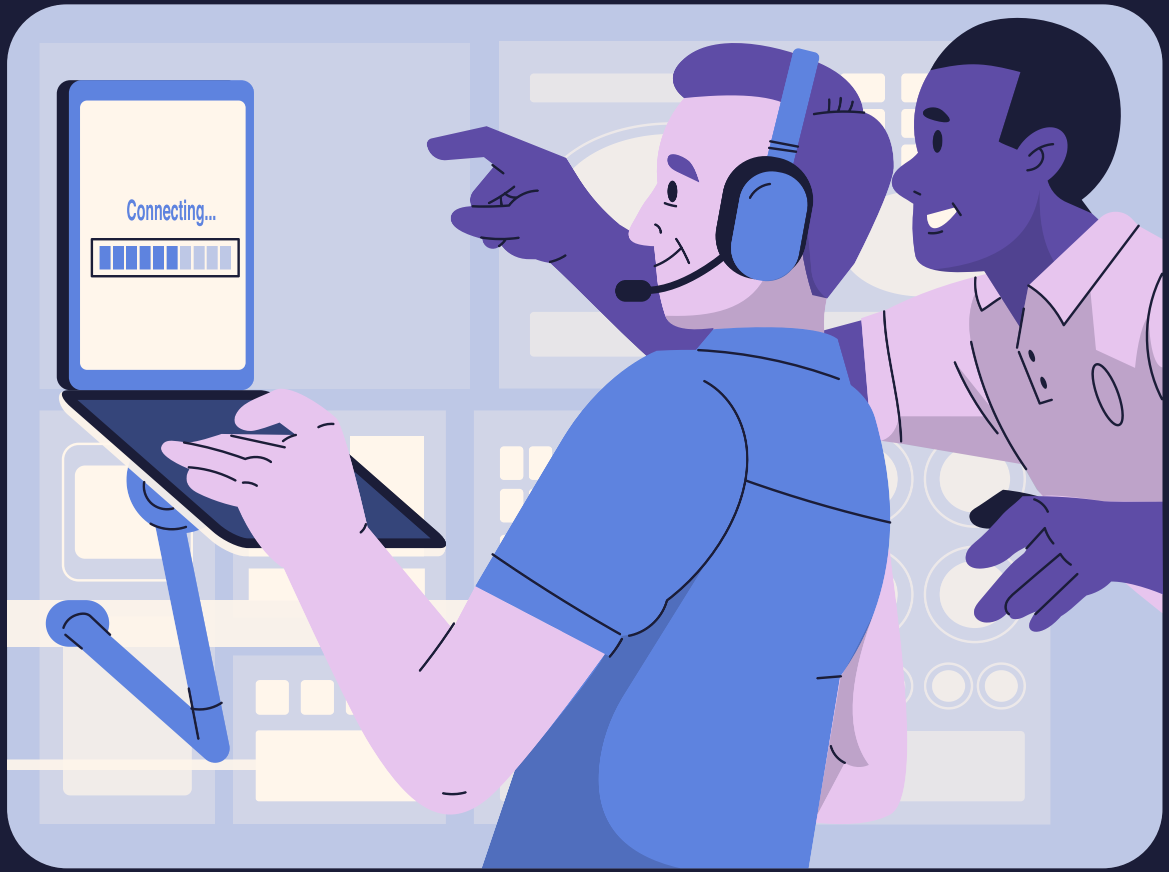
1. Introduction
2. Model and Dataset Information
3. Step 1: Loading the Dataset & the Tokenizer
4. Step 2: Preprocessing the Dataset
5. Step 3: Defining Training Arguments
6. Step 4: Fine-tuning the Model

1



INTRODUCTION

In this task, we will load, preprocess, and fine-tune a T5 model on a dataset of news articles for a summarization task.



2

MODEL AND DATASET INFORMATION

1 **MODEL CHECKPOINT:**

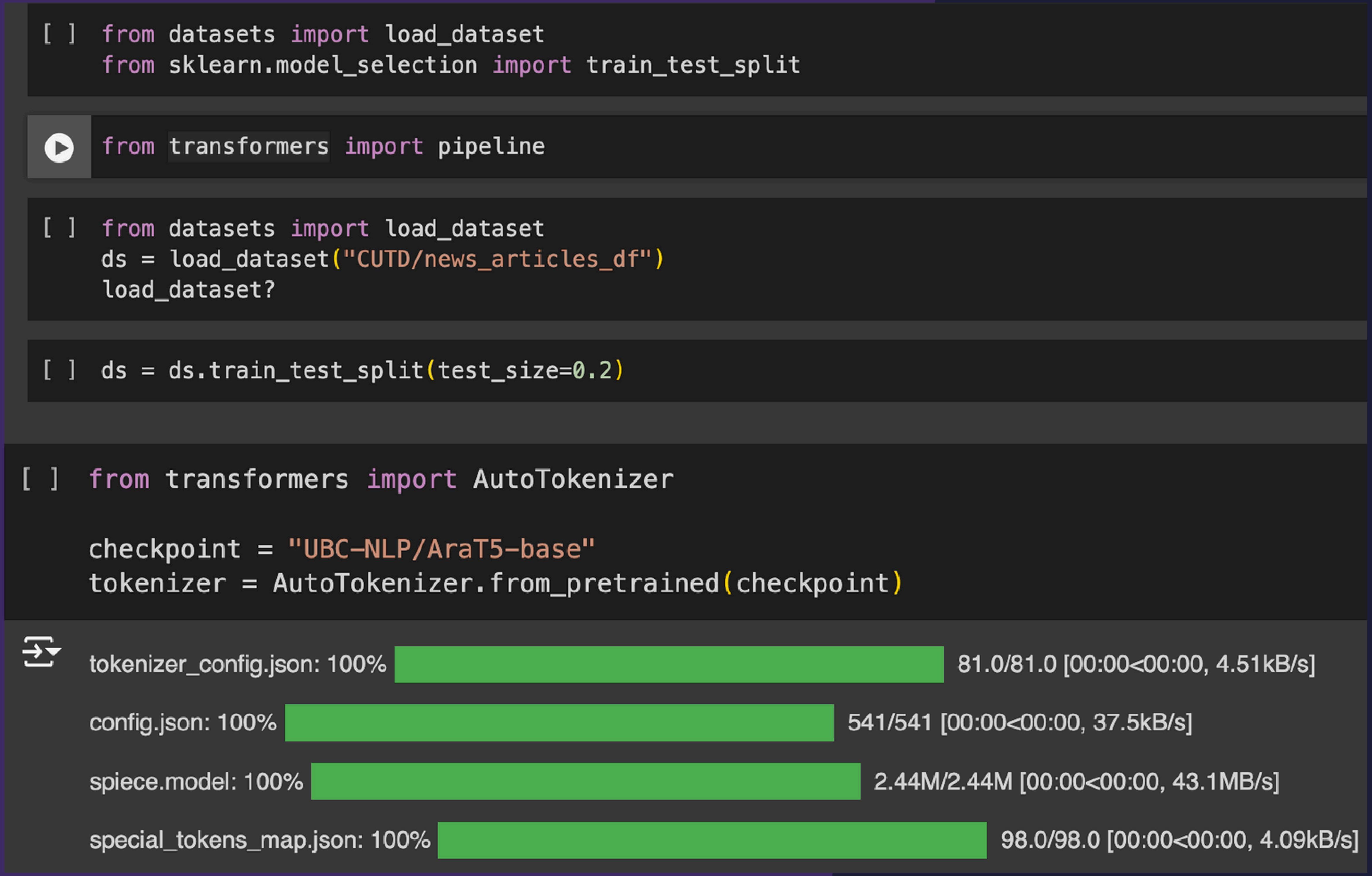
Used the pre-trained model checkpoint UBC-NLP/AraT5-base

2 **DATASET:**

Used the CUTD/news_articles_df dataset.

STEP 1:

LOADING THE DATASET & THE TOKENIZER



```
from datasets import load_dataset
from sklearn.model_selection import train_test_split
```

```
from transformers import pipeline
```

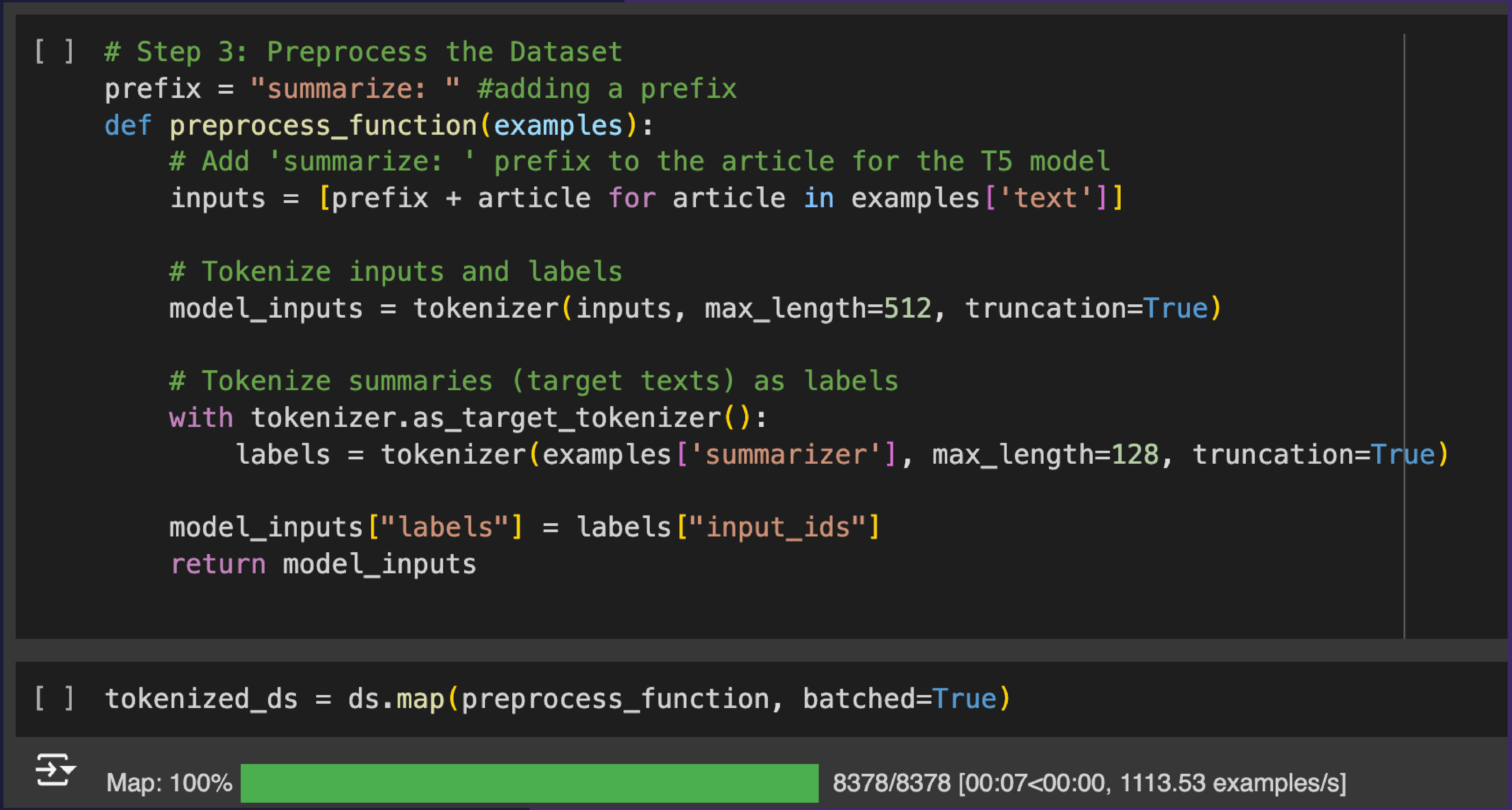
```
from datasets import load_dataset
ds = load_dataset("CUTD/news_articles_df")
load_dataset?
```

```
ds = ds.train_test_split(test_size=0.2)
```

```
from transformers import AutoTokenizer

checkpoint = "UBC-NLP/AraT5-base"
tokenizer = AutoTokenizer.from_pretrained(checkpoint)
```

```
tokenizer_config.json: 100% 81.0/81.0 [00:00<00:00, 4.51kB/s]
config.json: 100% 541/541 [00:00<00:00, 37.5kB/s]
spiece.model: 100% 2.44M/2.44M [00:00<00:00, 43.1MB/s]
special_tokens_map.json: 100% 98.0/98.0 [00:00<00:00, 4.09kB/s]
```

STEP 2: PREPROCESSING THE DATASET



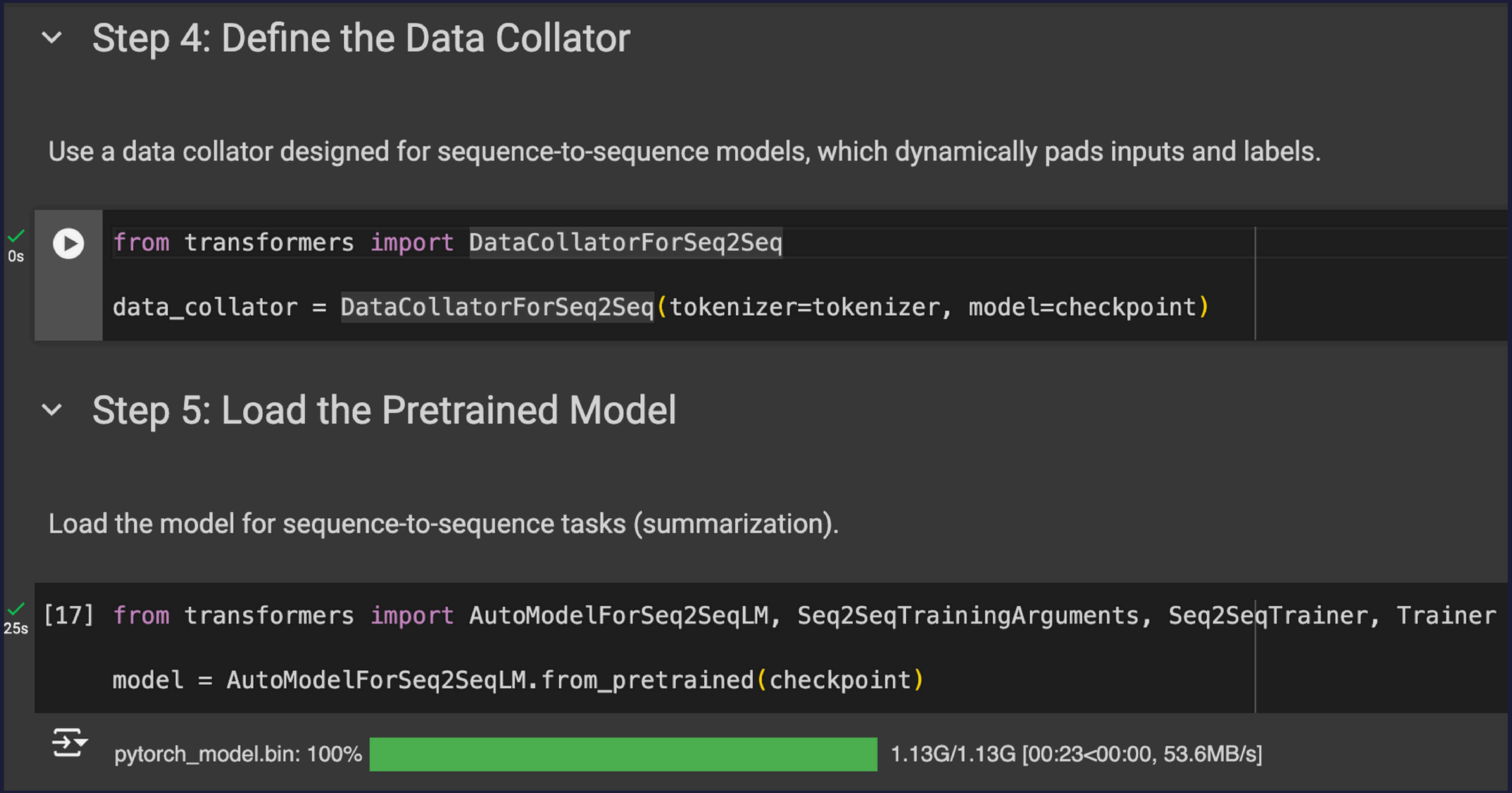
```
[ ] # Step 3: Preprocess the Dataset
    prefix = "summarize: " #adding a prefix
    def preprocess_function(examples):
        # Add 'summarize: ' prefix to the article for the T5 model
        inputs = [prefix + article for article in examples['text']]

        # Tokenize inputs and labels
        model_inputs = tokenizer(inputs, max_length=512, truncation=True)

        # Tokenize summaries (target texts) as labels
        with tokenizer.as_target_tokenizer():
            labels = tokenizer(examples['summarizer'], max_length=128, truncation=True)

        model_inputs["labels"] = labels["input_ids"]
        return model_inputs
```

```
[ ] tokenized_ds = ds.map(preprocess_function, batched=True)
```

```
Map: 100% 8378/8378 [00:07<00:00, 1113.53 examples/s]
```

STEP 3:

TRAINING



Step 4: Define the Data Collator

Use a data collator designed for sequence-to-sequence models, which dynamically pads inputs and labels.

0s

```
from transformers import DataCollatorForSeq2Seq

data_collator = DataCollatorForSeq2Seq(tokenizer=tokenizer, model=checkpoint)
```

Step 5: Load the Pretrained Model

Load the model for sequence-to-sequence tasks (summarization).

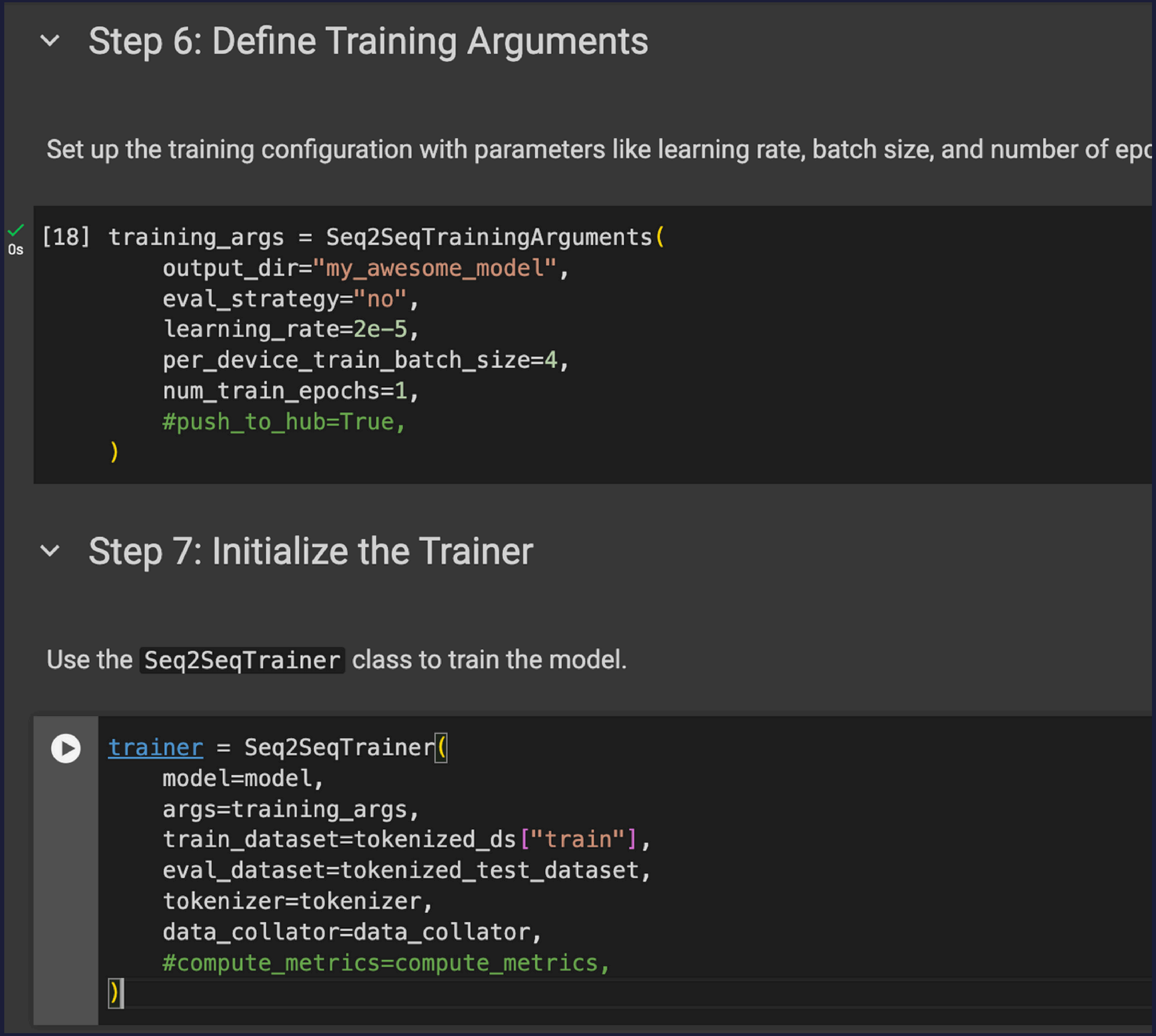
25s

```
[17] from transformers import AutoModelForSeq2SeqLM, Seq2SeqTrainingArguments, Seq2SeqTrainer, Trainer

     model = AutoModelForSeq2SeqLM.from_pretrained(checkpoint)
```

```
pytorch_model.bin: 100%  1.13G/1.13G [00:23<00:00, 53.6MB/s]
```

STEP 3:

TRAINING



Step 6: Define Training Arguments

Set up the training configuration with parameters like learning rate, batch size, and number of epc

```
[18] training_args = Seq2SeqTrainingArguments(
         output_dir="my_awesome_model",
         eval_strategy="no",
         learning_rate=2e-5,
         per_device_train_batch_size=4,
         num_train_epochs=1,
         #push_to_hub=True,
     )
```

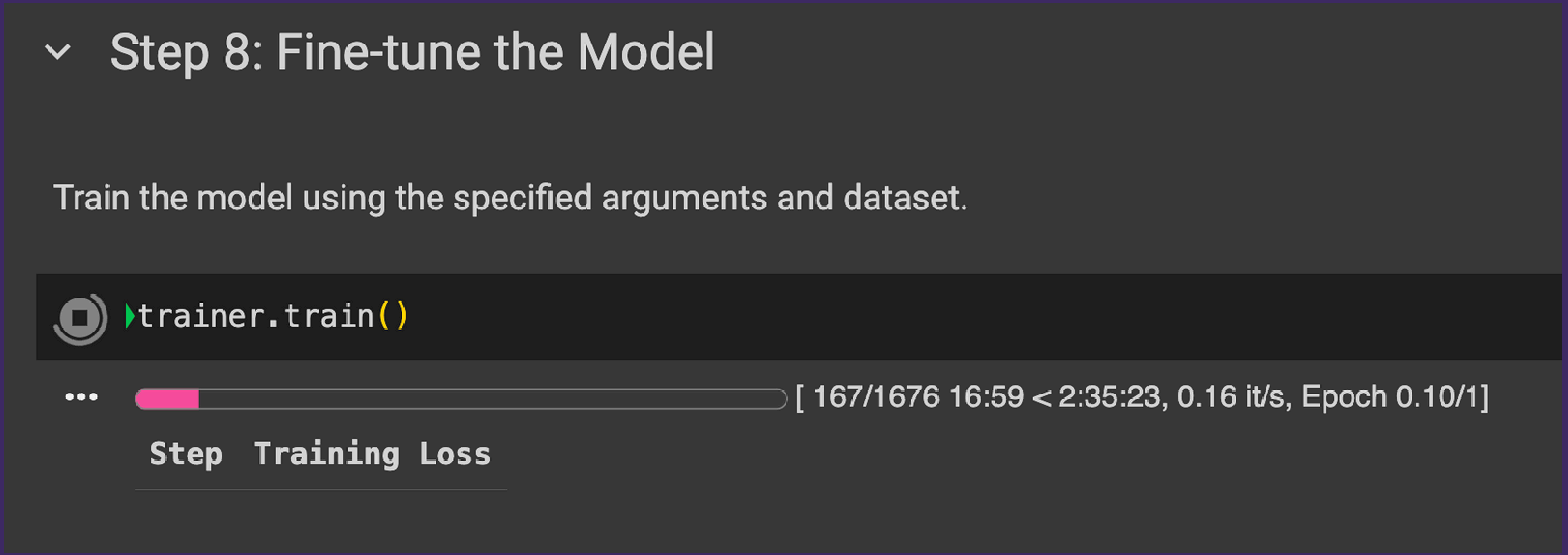
Step 7: Initialize the Trainer

Use the `Seq2SeqTrainer` class to train the model.

```
trainer = Seq2SeqTrainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_ds["train"],
    eval_dataset=tokenized_test_dataset,
    tokenizer=tokenizer,
    data_collator=data_collator,
    #compute_metrics=compute_metrics,
)
```

STEP 4:

FINE-TUNEING THE MODEL



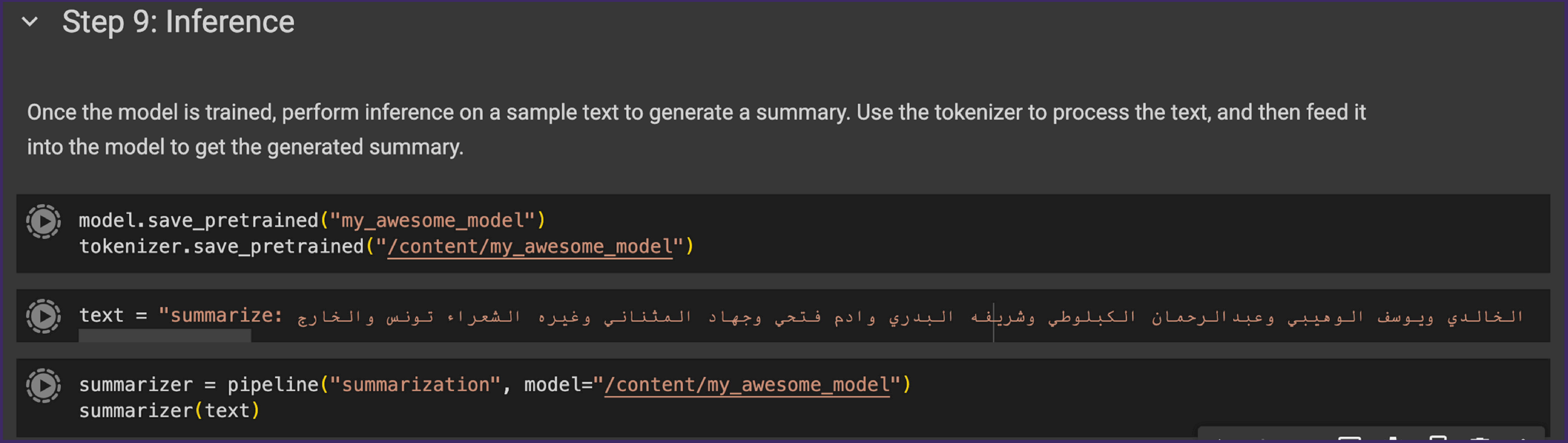
Step 8: Fine-tune the Model

Train the model using the specified arguments and dataset.

```
trainer.train()
```

```
[ 167/1676 16:59 < 2:35:23, 0.16 it/s, Epoch 0.10/1]
Step  Training Loss
```

CONCLUSION



Step 9: Inference

Once the model is trained, perform inference on a sample text to generate a summary. Use the tokenizer to process the text, and then feed it into the model to get the generated summary.

```
model.save_pretrained("my_awesome_model")
tokenizer.save_pretrained("/content/my_awesome_model")
```

```
text = "summarize: الخالدي ويوسف الوهيبي وعبدالرحمان الكبلوطي وشريفه البدري وادم فتحي وجهاد المثناني وغيره الشعراء تونس والخارج
```

```
summarizer = pipeline("summarization", model="/content/my_awesome_model")
summarizer(text)
```

THNAK YOU.