

```
# 📦 Install required libraries
!pip install transformers datasets scikit-learn --quiet
!pip install evaluate --quiet
```

```
🔄 _____ 491.4/491.4 kB 15.5 MB/s eta 0:00:00
_____ 116.3/116.3 kB 11.8 MB/s eta 0:00:00
_____ 193.6/193.6 kB 19.3 MB/s eta 0:00:00
_____ 143.5/143.5 kB 13.1 MB/s eta 0:00:00
_____ 194.8/194.8 kB 18.8 MB/s eta 0:00:00
```

```
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the fo
torch 2.6.0+cu124 requires nvidia-cublas-cu12==12.4.5.8; platform_system == "Linux" and platform_machine == "x86_64", but you have nvidia-cublas-
torch 2.6.0+cu124 requires nvidia-cuda-cupti-cu12==12.4.127; platform_system == "Linux" and platform_machine == "x86_64", but you have nvidia-cud
torch 2.6.0+cu124 requires nvidia-cuda-nvrtc-cu12==12.4.127; platform_system == "Linux" and platform_machine == "x86_64", but you have nvidia-cud
torch 2.6.0+cu124 requires nvidia-cuda-runtime-cu12==12.4.127; platform_system == "Linux" and platform_machine == "x86_64", but you have nvidia-c
torch 2.6.0+cu124 requires nvidia-cudnn-cu12==9.1.0.70; platform_system == "Linux" and platform_machine == "x86_64", but you have nvidia-cudnn-cu
torch 2.6.0+cu124 requires nvidia-cufft-cu12==11.2.1.3; platform_system == "Linux" and platform_machine == "x86_64", but you have nvidia-cufft-cu
torch 2.6.0+cu124 requires nvidia-curand-cu12==10.3.5.147; platform_system == "Linux" and platform_machine == "x86_64", but you have nvidia-curan
torch 2.6.0+cu124 requires nvidia-cusolver-cu12==11.6.1.9; platform_system == "Linux" and platform_machine == "x86_64", but you have nvidia-cusol
torch 2.6.0+cu124 requires nvidia-cuspars-cu12==12.3.1.170; platform_system == "Linux" and platform_machine == "x86_64", but you have nvidia-cus
torch 2.6.0+cu124 requires nvidia-nvjitlink-cu12==12.4.127; platform_system == "Linux" and platform_machine == "x86_64", but you have nvidia-nvji
gcsfs 2025.3.2 requires fsspec==2025.3.2, but you have fsspec 2025.3.0 which is incompatible.
```

```
_____ 84.0/84.0 kB 3.9 MB/s eta 0:00:00
```

```
import torch
torch.cuda.is_available()
```

```
🔄 True
```

```
# 📦 Import required packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os
import torch
from datasets import Dataset
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from transformers import AutoTokenizer, AutoModelForSequenceClassification, Trainer, TrainingArguments
```

```
# 📦 Load the Amazon Fine Food Reviews dataset
import pandas as pd
```

```
print("🔵 Loading dataset...")
df = pd.read_csv('/content/amazon_reviews.csv')

# 🛠️ Keep only 'Text' and 'Score' columns
df = df[['Text', 'Score']].rename(columns={'Text': 'text', 'Score': 'score'})

# 🔥 Map Sentiment (0: Negative, 1: Neutral, 2: Positive)
def map_sentiment(score):
    if score <= 2:
        return 0 # Negative
    elif score == 3:
        return 1 # Neutral
    else:
        return 2 # Positive



df['label'] = df['score'].apply(map_sentiment)



print(f"✅ Dataset loaded! Total samples: {len(df)}")
df.head()
```

↔️ 🔵 Loading dataset...
✅ Dataset loaded! Total samples: 568454

	text	score	label	📊
0	I have bought several of the Vitality canned d...	5	2	📊
1	Product arrived labeled as Jumbo Salted Peanut...	1	0	
2	This is a confection that has been around a fe...	4	2	
3	If you are looking for the secret ingredient i...	2	0	
4	Great taffy at a great price. There was a wid...	5	2	



```
# 📁 Use full dataset without sampling
print(f"✅ Using full dataset with {len(df)} reviews for fine-tuning")
df.head()
```

  Using full dataset with 568454 reviews for fine-tuning

	text	score	label	
0	I have bought several of the Vitality canned d...	5	2	
1	Product arrived labeled as Jumbo Salted Peanut...	1	0	
2	This is a confection that has been around a fe...	4	2	
3	If you are looking for the secret ingredient i...	2	0	
4	Great taffy at a great price. There was a wid...	5	2	

```
# 🔥 Load RoBERTa tokenizer
from transformers import AutoTokenizer

print("🔧 Loading RoBERTa tokenizer...")
tokenizer = AutoTokenizer.from_pretrained("cardiffnlp/twitter-roberta-base-sentiment")
print("✅ Loaded RoBERTa tokenizer...")
```

  Loading RoBERTa tokenizer...

/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 secret in your
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(
config.json: 100% 747/747 [00:00<00:00, 78.9kB/s]
vocab.json: 100% 899k/899k [00:00<00:00, 4.16MB/s]
merges.txt: 100% 456k/456k [00:00<00:00, 6.13MB/s]
special_tokens_map.json: 100% 150/150 [00:00<00:00, 18.4kB/s]
✅ Loaded RoBERTa tokenizer...

```
# 🥄 Split dataset into train and validation
from sklearn.model_selection import train_test_split



print("🔧 Splitting dataset into train and validation sets...")
train_texts, val_texts, train_labels, val_labels = train_test_split(
    df['text'].tolist(),
    df['label'].tolist(),
```

```

    test_size=0.2,
    random_state=42
)

print(f"✅ Training samples: {len(train_texts)}")
print(f"✅ Validation samples: {len(val_texts)}")

```


 Splitting dataset into train and validation sets...
 ✅ Training samples: 454763
 ✅ Validation samples: 113691

```

# 🖋 Define tokenization function
def tokenize(batch):
    return tokenizer(batch['text'], padding='max_length', truncation=True, max_length=128)


# 🛠 Format as Huggingface Datasets
from datasets import Dataset

train_dataset = Dataset.from_dict({'text': train_texts, 'label': train_labels})
val_dataset = Dataset.from_dict({'text': val_texts, 'label': val_labels})

# ✏ Tokenize the datasets
train_dataset = train_dataset.map(tokenize, batched=True)
val_dataset = val_dataset.map(tokenize, batched=True)

print("✅ Tokenization and dataset formatting complete.")

```

 Map: 100% 454763/454763 [02:31<00:00, 2561.14 examples/s]
 Map: 100% 113691/113691 [00:43<00:00, 2169.28 examples/s]
 ✅ Tokenization and dataset formatting complete.

```

# 🔥 Load RoBERTa model
from transformers import AutoModelForSequenceClassification

print("🛠 Loading RoBERTa model...")
model = AutoModelForSequenceClassification.from_pretrained(
    "cardiffnlp/twitter-roberta-base-sentiment",
    num_labels=3 # 3 classes: Negative, Neutral, Positive
)

```



⚠ Loading RoBERTa model...

Xet Storage is enabled for this repo, but the 'hf_xet' package is not installed. Falling back to regular HTTP download. For better performance, i
WARNING:huggingface_hub.file_download:Xet Storage is enabled for this repo, but the 'hf_xet' package is not installed. Falling back to regular HT

pytorch_model.bin: 100%

499M/499M [00:01<00:00, 310MB/s]

```
from transformers import TrainingArguments
```

```
training_args = TrainingArguments(  
    output_dir="./results",  
    per_device_train_batch_size=32,  
    per_device_eval_batch_size=32,  
    num_train_epochs=1,  
    warmup_steps=50,  
    weight_decay=0.01,  
    logging_dir="./logs",  
    save_steps=10000,  
    logging_steps=500,  
    report_to="none"  
)
```

```
# Define compute_metrics function  
from sklearn.metrics import accuracy_score  
import torch
```

```
def compute_metrics(eval_pred):  
    logits, labels = eval_pred  
    predictions = torch.argmax(torch.tensor(logits), axis=-1)  
    return {"accuracy": accuracy_score(labels, predictions)}
```

```
from transformers import Trainer
```

```
trainer = Trainer(  
    model=model,  
    args=training_args,  
    train_dataset=train_dataset,  
    eval_dataset=val_dataset,  
    compute_metrics=compute_metrics  
)
```

```
# ✅ Mount Google Drive
from google.colab import drive
drive.mount('/content/drive')

# 🔥 Start fine-tuning RoBERTa
trainer.train()

# 📈 Evaluate validation accuracy after training
results = trainer.evaluate()
print(f"✅ Final Validation Accuracy: {results['eval_accuracy'] * 100:.2f}%")

# 📁 Define save path in Google Drive
drive_model_dir = "/content/drive/MyDrive/roberta_finetuned/"
os.makedirs(drive_model_dir, exist_ok=True)

# 📦 Save model and tokenizer directly to Google Drive
model.save_pretrained(drive_model_dir)
tokenizer.save_pretrained(drive_model_dir)

# 📄 Save validation accuracy
accuracy = results['eval_accuracy']
with open(os.path.join(drive_model_dir, "accuracy.txt"), "w") as f:
    f.write(str(accuracy))

print(f"✅ Validation Accuracy Saved to Drive: {accuracy:.4f}")

# 🗜️ Zip and save to Google Drive
!zip -r "/content/drive/MyDrive/roberta_finetuned.zip" ./models/roberta_finetuned/

print("✅ Zipped model saved to Google Drive.")
```

Step	Training Loss
500	0.356100
1000	0.309600
1500	0.297500
2000	0.295300
2500	0.288200
3000	0.261400
3500	0.258800
4000	0.263900
4500	0.260100
5000	0.253800
5500	0.251800
6000	0.242900
6500	0.242500
7000	0.239400
7500	0.235100
8000	0.233100
8500	0.224400
9000	0.226200
9500	0.226900
10000	0.217900
10500	0.216900
11000	0.207000
11500	0.206900
12000	0.199500
12500	0.205400

13000	0.207100
13500	0.205300
14000	0.201100

 [1620/3553 05:25 < 06:28, 4.98 it/s]

 [3553/3553 11:51]

✓ Final Validation Accuracy: 92.92%

✓ Validation Accuracy Saved to Drive: 0.9292

zip warning: name not matched: ./models/roberta_finetuned/

zip error: Nothing to do! (try: zip -r /content/drive/MyDrive/roberta_finetuned.zip . -i ./models/roberta_finetuned/)

✓ Zipped model saved to Google Drive.

```
results = trainer.evaluate()
print(f"✓ Final Validation Accuracy: {results['eval_accuracy']*100:.2f}%")
```

...  [3553/3553 16:17]

```
model.save_pretrained("./models/roberta_finetuned/")
tokenizer.save_pretrained("./models/roberta_finetuned/")
```

```
with open('./models/roberta_finetuned/accuracy.txt', 'w') as f:
    f.write(str(results['eval_accuracy']))
```

```
!zip -r roberta_finetuned.zip ./models/roberta_finetuned/
from google.colab import files
files.download('roberta_finetuned.zip')
```

Start coding or [generate](#) with AI.