!pip install datasets

import tensorflow as tf  
from transformers import TFAutoModelForSequenceClassification, AutoTokenizer  
from datasets import load\_dataset  
import numpy as np  
from sklearn.metrics import accuracy\_score, precision\_recall\_fscore\_support

# Load the dataset  
dataset = load\_dataset("hate\_speech\_offensive")  
print(dataset['train'][0]) # Check a sample

{"model\_id":"46ff670a028242c4bb46e3ee344d7902","version\_major":2,"version\_minor":0}

{"model\_id":"8cc048c879454a92b1180f83a8e3d00f","version\_major":2,"version\_minor":0}

{"model\_id":"d7bc8bc9b016404eac671298a86c049e","version\_major":2,"version\_minor":0}

{'count': 3, 'hate\_speech\_count': 0, 'offensive\_language\_count': 0, 'neither\_count': 3, 'class': 2, 'tweet': "!!! RT @mayasolovely: As a woman you shouldn't complain about cleaning up your house. &amp; as a man you should always take the trash out..."}

# Initialize the tokenizer  
tokenizer = AutoTokenizer.from\_pretrained('distilbert-base-uncased')  
  
def preprocess\_function(examples):  
 return tokenizer(examples['tweet'], padding="max\_length", truncation=True, max\_length=128)  
  
# Tokenize the dataset  
tokenized\_datasets = dataset.map(preprocess\_function, batched=True)

{"model\_id":"13600898f88f40a79e0c652b759eba1f","version\_major":2,"version\_minor":0}

{"model\_id":"4a48e74e5de64c48ba91794ae1bc5b7c","version\_major":2,"version\_minor":0}

{"model\_id":"664754a0f64d46288f42368b799d30e2","version\_major":2,"version\_minor":0}

{"model\_id":"cfd6d99a6e574975b0b5109267328bb3","version\_major":2,"version\_minor":0}

{"model\_id":"305400982ce94cbaa04abbaabfed2245","version\_major":2,"version\_minor":0}

from sklearn.model\_selection import train\_test\_split  
  
# Load the dataset  
dataset = load\_dataset("hate\_speech\_offensive")  
  
# Split the 'train' dataset into training and validation sets  
train\_test\_split = dataset['train'].train\_test\_split(test\_size=0.2)  
train\_dataset = train\_test\_split['train']  
validation\_dataset = train\_test\_split['test']  
  
print(train\_dataset[0]) # Check a sample

{'count': 3, 'hate\_speech\_count': 0, 'offensive\_language\_count': 3, 'neither\_count': 0, 'class': 1, 'tweet': 'RT @AC\_Hussle: When yall having a good day &amp; she brings up a bitch from Twitter http://t.co/LQVLDxOwsH'}

# Tokenize the train and validation datasets  
tokenized\_train = train\_dataset.map(preprocess\_function, batched=True)  
tokenized\_validation = validation\_dataset.map(preprocess\_function, batched=True)  
  
# Convert to TensorFlow datasets  
tf\_train\_dataset = tokenized\_train.to\_tf\_dataset(  
 columns=['input\_ids', 'attention\_mask'],  
 label\_cols='class',  
 shuffle=True,  
 batch\_size=16  
)  
  
tf\_validation\_dataset = tokenized\_validation.to\_tf\_dataset(  
 columns=['input\_ids', 'attention\_mask'],  
 label\_cols='class',  
 shuffle=False,  
 batch\_size=16  
)

{"model\_id":"017a6bbc94a246d2bafad971b4d8f2a3","version\_major":2,"version\_minor":0}

{"model\_id":"acf548cba8a14c0fb2f811f60b251c6a","version\_major":2,"version\_minor":0}

# Load the pre-trained DistilBERT model  
model = TFAutoModelForSequenceClassification.from\_pretrained('distilbert-base-uncased', num\_labels=3)

{"model\_id":"bd5673ef74f243ff80e58d68e0eda05d","version\_major":2,"version\_minor":0}

# Compile the model  
optimizer = tf.keras.optimizers.Adam(learning\_rate=2e-5)  
loss = tf.keras.losses.SparseCategoricalCrossentropy(from\_logits=True)  
model.compile(optimizer=optimizer, loss=loss, metrics=['accuracy'])

# Train the model  
# Train the model  
history = model.fit(tf\_train\_dataset, validation\_data=tf\_validation\_dataset, epochs=3)

Epoch 1/3  
1240/1240 [==============================] - 289s 233ms/step - loss: 0.2373 - accuracy: 0.9155 - val\_loss: 0.2538 - val\_accuracy: 0.9126  
Epoch 2/3  
1240/1240 [==============================] - 268s 216ms/step - loss: 0.1858 - accuracy: 0.9327 - val\_loss: 0.2612 - val\_accuracy: 0.9098  
Epoch 3/3  
1240/1240 [==============================] - 268s 216ms/step - loss: 0.1433 - accuracy: 0.9484 - val\_loss: 0.2977 - val\_accuracy: 0.8840

# Evaluate the model  
# Evaluate the model on the validation dataset  
results = model.evaluate(tf\_validation\_dataset)  
print(f"Validation Loss: {results[0]}, Validation Accuracy: {results[1]}")

310/310 [==============================] - 23s 73ms/step - loss: 0.2977 - accuracy: 0.8840  
Validation Loss: 0.29765763878822327, Validation Accuracy: 0.8840024471282959

import numpy as np  
import matplotlib.pyplot as plt  
# Plot training and validation loss  
plt.figure(figsize=(12, 5))  
plt.subplot(1, 2, 1)  
plt.plot(history.history['loss'], label='Train Loss')  
plt.plot(history.history['val\_loss'], label='Validation Loss')  
plt.title('Training and Validation Loss')  
plt.xlabel('Epochs')  
plt.ylabel('Loss')  
plt.legend()  
  
# Plot training and validation accuracy  
plt.subplot(1, 2, 2)  
plt.plot(history.history['accuracy'], label='Train Accuracy')  
plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')  
plt.title('Training and Validation Accuracy')  
plt.xlabel('Epochs')  
plt.ylabel('Accuracy')  
plt.legend()  
  
plt.show()

