Generative AI for NLP Program

Project

GA-NLP Mid-Term Project: Financial Product Complaint Classification and Summarization

Business Context

In today's financial landscape, customer complaints are pivotal for financial institutions, highlighting areas of dissatisfaction and guiding business improvements. The intricate task of classifying these complaints into specific product categories is crucial for understanding customer issues and enhancing service delivery. By employing Generative AI for text classification, businesses can gain a detailed understanding of customer grievances related to various financial products such as credit reports, student loans, and money transfers.

The integration of machine learning algorithms has revolutionized the automation of customer complaint classification. Utilizing these advanced techniques, financial institutions can swiftly and accurately categorize new complaints based on their content. This automation not only saves time and resources but also ensures timely responses to customer issues, thereby improving customer satisfaction and compliance with regulatory standards.

Additionally, this project will explore the summarization of customer narratives to provide more personalized solutions to complaints. By using Generative AI, businesses can enhance their ability to classify complaints more precisely and generate complaint summaries that facilitate more

tailored and effective service responses.

Embarking on this project of Financial Product Complaint Classification and Summarization, with a focus on classification and summarization accuracy, equips you with essential skills applicable to real-world business contexts. Through hands-on experience with code and implementation specifics, you'll gain the proficiency to build such solutions using open-source machine learning algorithms. This experience will serve as a compelling Proof-of-Concept, paving the way for the implementation of these advanced solutions in financial institutions.

Project Objective

Develop a Generative AI application using a Large Language Model to automate product classification and narrative summarization. This application will predict product categories, generate responses based on customer sentiment, and summarize narratives for the mediation team. We have been tackling this task with BERT and prompt engineering with LLMs. We will explore various techniques and select the most effective method.

Section 1 BERT Fine Tuning (10 Marks)

Question 1: Installling the necessary packages and importing libraries (1 Mark)

In [4]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [2]:
```
# install the libraries to load transformers==4.42.4 models
! pip install transformers==4.42.4
```

```
Requirement already satisfied: transformers==4.42.4 in /usr/local/lib/python3.10/dist-packages (4.42.4)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from transformers==4.42.4) (3.15.4)
Requirement already satisfied: huggingface-hub<1.0,>=0.23.2 in /usr/local/lib/python3.10/dist-packages (from transformers==4.42.
4) (0.23.5)
Requirement already satisfied: numpy<2.0,>=1.17 in /usr/local/lib/python3.10/dist-packages (from transformers==4.42.4) (1.26.4)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from transformers==4.42.4) (24.1)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from transformers==4.42.4) (6.0.2)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-packages (from transformers==4.42.4) (2024.5.
15)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from transformers==4.42.4) (2.32.3)
Requirement already satisfied: safetensors>=0.4.1 in /usr/local/lib/python3.10/dist-packages (from transformers==4.42.4) (0.4.4)
Requirement already satisfied: tokenizers<0.20,>=0.19 in /usr/local/lib/python3.10/dist-packages (from transformers==4.42.4) (0.
19.1)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-packages (from transformers==4.42.4) (4.66.5)
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.23.2->t
ransformers==4.42.4) (2024.6.1)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>
=0.23.2->transformers==4.42.4) (4.12.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->transformers=
=4.42.4) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->transformers==4.42.4) (3.
7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->transformers==4.42.
4) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->transformers==4.42.
4) (2024.7.4)
```

In [2]:

```python
# Import necessary libraries for data manipulation and analysis
import pandas as pd
import numpy as np

# Import visualization libraries
import seaborn as sns
import matplotlib.pyplot as plt

# Import modules from scikit-learn for machine learning tasks
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusion_matrix, f1_score, precision_score, recall_score, accuracy_score, classification_report

# Import TensorFlow for deep learning tasks
import tensorflow as tf

import re
```

```
import json
import os  # For interacting with the operating system
import sys  # For accessing system-specific parameters and functions
import glob  # For file pattern matching
import locale
```

In [3]:
```
#from datasets import load_dataset
from collections import Counter
from tqdm import tqdm
from huggingface_hub import hf_hub_download
#from llama_cpp import Llama
```

In [4]:
```
# Import BertTokenizer, TFBertForSequenceClassification from the Hugging Face transformers library
from transformers import BertTokenizer, TFBertForSequenceClassification
```

In [5]:
```
# Set the seed for the TensorFlow random number generator to ensure reproducibility
tf.random.set_seed(42)
```

Question 2: Data preprocessing for Bert Fine Tunning (2 Marks)

In [2]:
```
# Load a CSV File containing Dataset of 500 products, narrative and summary (summary of narrative)
data=pd.read_csv(r'/content/Complains_classification - Copy.csv')
```

In [3]:
```
Bert_data=data[['product','narrative']]
```

In [9]:
```
Bert_data.head()
```

Out[9]:

	product	narrative
0	credit_card	purchase order day shipping amount receive pro...
1	credit_card	forwarded message date tue subject please inve...
2	retail_banking	forwarded message cc sent friday pdt subject f...
3	credit_reporting	payment history missing credit report speciali...
4	credit_reporting	payment history missing credit report made mis...

doubt sir do we require to remove duplicate row in this case?

In [ ]:
```
#lets check any duplicated values in original data set
data.duplicated().sum()
```

Out[ ]:
```
0
```

In [ ]:
```
#lets check any duplicated values are there or not
print("the data has ",Bert_data.duplicated().sum()," number of duplicated values")
```

```
the data has  260  number of duplicated values
```

In [ ]:
```
#lets check any missing values in data set
print("the number of missing values are", Bert_data.isnull().sum())
```

```
the number of missing values are product      0
narrative    0
dtype: int64
```

In [10]:
```
#lets make lower string all data present in product
Bert_data.loc[:,'product'] = Bert_data['product'].str.lower()
Bert_data.loc[:, 'narrative'] = Bert_data['narrative'].str.lower()
```

In [11]:
```
Bert_data.head()
```

Out[11]:

	product	narrative
0	credit_card	purchase order day shipping amount receive pro...
1	credit_card	forwarded message date tue subject please inve...
2	retail_banking	forwarded message cc sent friday pdt subject f...
3	credit_reporting	payment history missing credit report speciali...
4	credit_reporting	payment history missing credit report made mis...

In [12]:
```
#removing punctuation from text in each row
Bert_data.loc[:,'product'] = Bert_data['product'].str.replace(r'[^\w\s]', '', regex=True)
Bert_data.loc[:,'narrative'] = Bert_data['narrative'].str.replace(r'[^\w\s]', '', regex=True)
```

In [13]:
```
Bert_data.head()
```

Out[13]:

	product	narrative
0	credit_card	purchase order day shipping amount receive pro...
1	credit_card	forwarded message date tue subject please inve...
2	retail_banking	forwarded message cc sent friday pdt subject f...
3	credit_reporting	payment history missing credit report speciali...
4	credit_reporting	payment history missing credit report made mis...

In [14]:
```
# creating dependent and independent variable from Bert_data
#here narrative will be the input varibale that we will give input to model
train_test = Bert_data['narrative']
#here product will be target variable
y = Bert_data['product']
```

In [15]:
```
X_train,X_test,y_train,y_test=train_test_split(train_test,y,test_size=0.2,random_state=42)
```

In [ ]:
```

```

In [16]:
```
label_encoder = LabelEncoder()
y_train_enc = label_encoder.fit_transform(y_train)
y_test_enc = label_encoder.fit_transform(y_test)
```

In [ ]:
```
y_train_enc
```

Out[ ]:
```
array([1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 2, 3, 1, 1, 1, 0, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 4, 1, 1, 0, 1, 1, 1, 1, 1, 3, 3, 1, 0, 1, 1, 2,
       1, 1, 3, 1, 1, 1, 1, 0, 4, 3, 3, 1, 1, 1, 1, 3, 2, 1, 1, 3, 0, 1,
       3, 1, 0, 2, 2, 1, 1, 0, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 4, 3, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 4, 1, 1, 1, 1, 3, 1, 3, 1, 4,
       2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 4, 3, 1, 2, 1, 1, 1, 4, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 3, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 3, 1, 1, 1, 1, 2, 1, 1, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 4, 1, 1, 2, 1, 1,
       1, 4, 1, 1, 3, 1, 1, 1, 1, 1, 1, 0, 4, 3, 1, 1, 1, 1, 1, 2, 2, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 2, 1, 1, 1, 1, 1, 1, 1, 0,
       0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 1, 2, 1, 1, 1, 1, 3,
       1, 2, 1, 1, 1, 1, 1, 1, 1, 4, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 4, 3, 1, 0, 1, 1, 0, 1, 3, 1, 4, 1, 1,
       1, 2, 1, 1, 3, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 0,
       1, 4, 0, 1, 1, 1, 2, 1, 1, 3, 3, 1, 1, 1, 1, 1, 2, 1, 1, 3, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 4, 2, 1, 1, 1, 1, 1, 1, 1, 0, 1,
       1, 1, 1, 0])
```

In [17]:
```
y_test_enc
```

Out[17]:
```
array([1, 3, 1, 1, 1, 1, 1, 1, 1, 3, 1, 1, 1, 4, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 2, 0, 1, 1, 0, 1, 1, 1, 1, 1, 4, 1, 1, 1, 1, 4, 1, 0, 1, 1,
       1, 3, 0, 0, 1, 3, 1, 1, 1, 1, 1, 2, 1, 3, 1, 3, 1, 1, 1, 1, 2, 1,
       4, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 2, 1, 1, 1, 2, 1, 1,
       1, 1, 1, 1, 1, 1, 2, 1, 3, 2, 1, 3])
```

Question 3: Tokenization (1 Mark)

In [18]:
```
# loading and creating an instance of the BERT tokenizer
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
# specifying the maximum length of the input 512
max_length = 512
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:89: 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 Google Colab and restart your session.
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:   0%|          | 0.00/48.0 [00:00<?, ?B/s]
```

```
vocab.txt:   0%|          | 0.00/232k [00:00<?, ?B/s]
tokenizer.json:   0%|          | 0.00/466k [00:00<?, ?B/s]
config.json:   0%|          | 0.00/570 [00:00<?, ?B/s]
```

In [19]:
```python
X_train_tokenized = tokenizer(
    X_train.values.tolist(),    # passing the data as a list to the tokenizer
    max_length=max_length,    # specifies the maximum length of the tokenized data
    padding='max_length',    # padding the data to the specified maximum length
    truncation=True,    # truncating the input if it is longer than the specified maximum length
    return_attention_mask=True,    # specifying to return attention masks
    return_tensors='tf',    # specifying to return the output as tensorflow tensors
)
X_test_tokenized = tokenizer(
    X_test.values.tolist(),
    max_length=max_length,
    padding='max_length',
    truncation=True,
    return_attention_mask=True,
    return_tensors='tf',
)
```

Question 4: Creating Tensorflow dataset (1 Mark)

In [20]:
```python
batch_size = 8
```

In [21]:
```python
# Convert the tokenized input and output into a batched TensorFlow dataset for training
train_tokenized_tf = tf.data.Dataset.from_tensor_slices((
    {
        'input_ids': X_train_tokenized['input_ids'],
        'attention_mask': X_train_tokenized['attention_mask']
    }, y_train_enc
)).batch(batch_size)

# Convert the tokenized input and output into a batched TensorFlow dataset for testing
test_tokenized_tf = tf.data.Dataset.from_tensor_slices((
    {
        'input_ids': X_test_tokenized['input_ids'],
        'attention_mask': X_test_tokenized['attention_mask']
    }, y_test_enc
)).batch(batch_size)
```

Question 5 Evaluating the base model's performance in product classification.(1 Marks)

In [38]:
```
def bert_f1_score(actual_val,perdicted_val):
    micro_f1_score = f1_score(actual_val, perdicted_val, average="micro")
    return micro_f1_score
```

In [23]:
```
# Actual product class
actual_val = np.concatenate([y for x, y in test_tokenized_tf], axis=0)
```

In [24]:
```
num_classes = y.nunique()
# Initialize Model using BERT for sequence classification
model = TFBertForSequenceClassification.from_pretrained('bert-base-uncased', num_labels=num_classes)
```

```
model.safetensors:   0%|          | 0.00/440M [00:00<?, ?B/s]
All PyTorch model weights were used when initializing TFBertForSequenceClassification.

Some weights or buffers of the TF 2.0 model TFBertForSequenceClassification were not initialized from the PyTorch model and are
newly initialized: ['classifier.weight', 'classifier.bias']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
```

In [ ]:
```
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=2e-5),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

In [ ]:
```
model.fit(train_tokenized_tf,
          validation_data=test_tokenized_tf,
          epochs=3,
          batch_size=batch_size)
```

```
Epoch 1/3
50/50 [==============================] - 3392s 67s/step - loss: 1.1890 - accuracy: 0.7475 - val_loss: 0.8379 - val_accuracy: 0.7
300
Epoch 2/3
50/50 [==============================] - 3373s 67s/step - loss: 1.0485 - accuracy: 0.7400 - val_loss: 0.7826 - val_accuracy: 0.7
400
Epoch 3/3
50/50 [==============================] - 3360s 67s/step - loss: 0.6221 - accuracy: 0.7950 - val_loss: 1.0087 - val_accuracy: 0.7
400
```

Out[ ]:
```
<tf_keras.src.callbacks.History at 0x7f07b5ff4e50>
```

In [ ]:
```
# Make prediction on test_tokenized_tf
preds_raw_test = model.predict(test_tokenized_tf)
preds_test = np.argmax(np.array(tf.nn.softmax(preds_raw_test.logits)), axis=1)
```

```
13/13 [==============================] - 238s 18s/step
```

In [ ]:
```
# Evaluate bert base model
from sklearn.metrics import f1_score
f1_score= bert_f1_score(actual_val, preds_test)
print(f1_score)
```

```
0.74
```

Question 6 Fine-Tuning Bert Model on training set (2 Marks)

In [ ]:
```
num_classes = y.nunique()
# Model initialization using BERT for sequence classification
model =  TFBertForSequenceClassification.from_pretrained('bert-base-uncased', num_labels=num_classes)
```

```
All PyTorch model weights were used when initializing TFBertForSequenceClassification.

Some weights or buffers of the TF 2.0 model TFBertForSequenceClassification were not initialized from the PyTorch model and are
newly initialized: ['classifier.weight', 'classifier.bias']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
```

In [ ]:
```
# setting the learning rate for the optimizer
learning_rate = 1e-5

# Setting the optimizer to Adam
optimizer = tf.keras.optimizers.Adam(learning_rate=learning_rate, epsilon=1e-08)

# Specify the loss function for the model
loss = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)

# Define evaluation metric(s) for the model
metric = [tf.keras.metrics.SparseCategoricalAccuracy('accuracy')]

# Compile the model with the chosen optimizer, loss function, and metrics
model.compile(optimizer=optimizer,
              loss=loss,
              metrics=metric)
```

In [ ]:
```
# Calculate class weights for imbalanced dataset
cw = (y_train_enc.shape[0]) / np.bincount(y_train_enc)

# Create a dictionary mapping class indices to their respective class weights
cw_dict = {}
for i in range(cw.shape[0]):
    cw_dict[label_encoder.transform(label_encoder.classes_)[i]] = cw[i]
```

In [ ]:
```
# Number of training epochs
n_epochs = 1
#train bert model
bert_base_tuned = model.fit(train_tokenized_tf,
          validation_data=test_tokenized_tf,
          epochs=n_epochs,
          batch_size=batch_size,
          class_weight=cw_dict)
```

```
50/50 [==============================] - 3499s 69s/step - loss: 7.2895 - accuracy: 0.6650 - val_loss: 0.9339 - val_accuracy: 0.7
300
```

Question 7. Evaluating the trained model performance (1 Mark)

In [ ]:
```
# Generate raw predictions on the test dataset using the trained model
preds_raw_val =  model.predict(test_tokenized_tf)

# Extract predicted labels by finding the index with the highest probability for each example
preds_val = np.argmax(np.array(tf.nn.softmax(preds_raw_val.logits)), axis=1)
```

```
13/13 [==============================] - 250s 19s/step
```

In [ ]:
```
# Evaluate bert trained model
from sklearn.metrics import f1_score
f1_score= f1_score(actual_val, preds_val, average='micro')
print(f1_score)
```

```
0.7299999999999999
```

Question 8: Write your observations (1 Mark)

Observations: Learning Rate Impact:

Original Learning Rate (2e-5): Provided a good balance, leading to an F1 score of 0.74 and a loss of 1.24. Tweaked Learning Rate (1e-5): Resulted in lower accuracy (0.72) and higher loss (7.01). A lower learning rate might cause the model to converge more slowly, leading to insufficient training or poor convergence within the given epochs. Loss Function and Metrics:

The loss function and metrics you used (SparseCategoricalCrossentropy and SparseCategoricalAccuracy) are appropriate for classification tasks. However, if the learning rate is too low, the model might not effectively minimize the loss. Possible Reasons for Performance Decrease: Learning Rate Too Low:

A very low learning rate can cause the model to take very small steps during training, resulting in slower convergence or getting stuck in local minima. This can lead to higher loss and lower accuracy if the model doesn't effectively learn from the data. Training Epochs:

If the number of epochs was not increased, the model might not have had enough time to converge with the lower learning rate. Consider increasing the number of epochs to give the model more time to learn. Overfitting or Underfitting:

With a lower learning rate, the model might underfit the data if it's not complex enough or if it hasn't trained sufficiently. It might not have had enough capacity to learn the patterns in the data. Recommendations: Adjust Learning Rate:

Experiment with intermediate learning rates between 1e-5 and 2e-5 to find a better balance. You might also use learning rate schedules to adjust the learning rate dynamically during training. Increase Training Time:

Try increasing the number of epochs to allow the model to train longer with the new learning rate. Monitor Training:

Track training and validation losses and accuracies during training to identify if and when the model starts to overfit or underfit. Hyperparameter Tuning:

Perform hyperparameter tuning to find the optimal learning rate, batch size, and other parameters for your specific model and dataset. Learning Rate Schedulers:

Use learning rate schedulers like ReduceLROnPlateau or ExponentialDecay to adjust the learning rate during training based on model performance. By carefully adjusting and experimenting with these parameters, you can improve your model’s performance.

Prompt Engineering

Section 2: Install Libraries for Prompt Engineering and Setting up Mistral Model (3 Marks)

Question 9: Install neccessary libraries (1 Mark)

In [7]:
```
# Installation for GPU llama_cpp_python==0.2.28 # since GPU is not working properly taking it to system GPU
! pip install llama_cpp_python==0.2.28
# For downloading the models from HF Hub huggingface-hub==0.23.2
! pip install huggingface-hub==0.23.2
# install evaluate==0.4.1 and bert-score==0.3.13 using pip command
! pip install evaluate==0.4.1 bert-score==0.3.13
# install numpy==1.25.2
#! pip install numpy==1.25.2
```

```
Requirement already satisfied: llama_cpp_python==0.2.28 in /usr/local/lib/python3.10/dist-packages (0.2.28)
Requirement already satisfied: typing-extensions>=4.5.0 in /usr/local/lib/python3.10/dist-packages (from llama_cpp_python==0.2.2
8) (4.12.2)
Requirement already satisfied: numpy>=1.20.0 in /usr/local/lib/python3.10/dist-packages (from llama_cpp_python==0.2.28) (1.26.4)
Requirement already satisfied: diskcache>=5.6.1 in /usr/local/lib/python3.10/dist-packages (from llama_cpp_python==0.2.28) (5.6.
3)
Requirement already satisfied: huggingface-hub==0.23.2 in /usr/local/lib/python3.10/dist-packages (0.23.2)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from huggingface-hub==0.23.2) (3.15.4)
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub==0.23.2) (2024.
6.1)
Requirement already satisfied: packaging>=20.9 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub==0.23.2) (24.1)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub==0.23.2) (6.0.2)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from huggingface-hub==0.23.2) (2.32.3)
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub==0.23.2) (4.66.5)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub==0.2
3.2) (4.12.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->huggingface-h
ub==0.23.2) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->huggingface-hub==0.23.2)
(3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->huggingface-hub==0.
23.2) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->huggingface-hub==0.
23.2) (2024.7.4)
Collecting evaluate==0.4.1
  Using cached evaluate-0.4.1-py3-none-any.whl.metadata (9.4 kB)
Collecting bert-score==0.3.13
  Using cached bert_score-0.3.13-py3-none-any.whl.metadata (15 kB)
Collecting datasets>=2.0.0 (from evaluate==0.4.1)
  Using cached datasets-2.20.0-py3-none-any.whl.metadata (19 kB)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from evaluate==0.4.1) (1.26.4)
Collecting dill (from evaluate==0.4.1)
  Using cached dill-0.3.8-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from evaluate==0.4.1) (2.1.4)
Requirement already satisfied: requests>=2.19.0 in /usr/local/lib/python3.10/dist-packages (from evaluate==0.4.1) (2.32.3)
Requirement already satisfied: tqdm>=4.62.1 in /usr/local/lib/python3.10/dist-packages (from evaluate==0.4.1) (4.66.5)
Collecting xxhash (from evaluate==0.4.1)
  Using cached xxhash-3.4.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (12 kB)
Collecting multiprocess (from evaluate==0.4.1)
  Using cached multiprocess-0.70.16-py310-none-any.whl.metadata (7.2 kB)
Requirement already satisfied: fsspec>=2021.05.0 in /usr/local/lib/python3.10/dist-packages (from fsspec[http]>=2021.05.0->evalu
ate==0.4.1) (2024.6.1)
Requirement already satisfied: huggingface-hub>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from evaluate==0.4.1) (0.23.2)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from evaluate==0.4.1) (24.1)
```

```
Collecting responses<0.19 (from evaluate==0.4.1)
  Using cached responses-0.18.0-py3-none-any.whl.metadata (29 kB)
Requirement already satisfied: torch>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from bert-score==0.3.13) (2.3.1+cu121)
Requirement already satisfied: transformers>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from bert-score==0.3.13) (4.42.4)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from bert-score==0.3.13) (3.7.1)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from datasets>=2.0.0->evaluate==0.4.1) (3.1
5.4)
Collecting pyarrow>=15.0.0 (from datasets>=2.0.0->evaluate==0.4.1)
  Using cached pyarrow-17.0.0-cp310-cp310-manylinux_2_28_x86_64.whl.metadata (3.3 kB)
Requirement already satisfied: pyarrow-hotfix in /usr/local/lib/python3.10/dist-packages (from datasets>=2.0.0->evaluate==0.4.1)
(0.6)
Collecting fsspec>=2021.05.0 (from fsspec[http]>=2021.05.0->evaluate==0.4.1)
  Using cached fsspec-2024.5.0-py3-none-any.whl.metadata (11 kB)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-packages (from datasets>=2.0.0->evaluate==0.4.1) (3.10.
1)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from datasets>=2.0.0->evaluate==0.4.1)
(6.0.2)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.7.
0->evaluate==0.4.1) (4.12.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas->evaluate==0.4.1)
(2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas->evaluate==0.4.1) (2024.1)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas->evaluate==0.4.1) (2024.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->evalu
ate==0.4.1) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->evaluate==0.4.1)
(3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->evaluate==
0.4.1) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->evaluate==
0.4.1) (2024.7.4)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch>=1.0.0->bert-score==0.3.13) (1.13.1)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch>=1.0.0->bert-score==0.3.13) (3.3)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch>=1.0.0->bert-score==0.3.13) (3.1.4)
Collecting nvidia-cuda-nvrtc-cu12==12.1.105 (from torch>=1.0.0->bert-score==0.3.13)
  Using cached nvidia_cuda_nvrtc_cu12-12.1.105-py3-none-manylinux1_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-runtime-cu12==12.1.105 (from torch>=1.0.0->bert-score==0.3.13)
  Using cached nvidia_cuda_runtime_cu12-12.1.105-py3-none-manylinux1_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-cupti-cu12==12.1.105 (from torch>=1.0.0->bert-score==0.3.13)
  Using cached nvidia_cuda_cupti_cu12-12.1.105-py3-none-manylinux1_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cudnn-cu12==8.9.2.26 (from torch>=1.0.0->bert-score==0.3.13)
  Using cached nvidia_cudnn_cu12-8.9.2.26-py3-none-manylinux1_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cublas-cu12==12.1.3.1 (from torch>=1.0.0->bert-score==0.3.13)
  Using cached nvidia_cublas_cu12-12.1.3.1-py3-none-manylinux1_x86_64.whl.metadata (1.5 kB)
```

```
Collecting nvidia-cufft-cu12==11.0.2.54 (from torch>=1.0.0->bert-score==0.3.13)
  Using cached nvidia_cufft_cu12-11.0.2.54-py3-none-manylinux1_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-curand-cu12==10.3.2.106 (from torch>=1.0.0->bert-score==0.3.13)
  Using cached nvidia_curand_cu12-10.3.2.106-py3-none-manylinux1_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cusolver-cu12==11.4.5.107 (from torch>=1.0.0->bert-score==0.3.13)
  Using cached nvidia_cusolver_cu12-11.4.5.107-py3-none-manylinux1_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cusparse-cu12==12.1.0.106 (from torch>=1.0.0->bert-score==0.3.13)
  Using cached nvidia_cusparse_cu12-12.1.0.106-py3-none-manylinux1_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-nccl-cu12==2.20.5 (from torch>=1.0.0->bert-score==0.3.13)
  Using cached nvidia_nccl_cu12-2.20.5-py3-none-manylinux2014_x86_64.whl.metadata (1.8 kB)
Collecting nvidia-nvtx-cu12==12.1.105 (from torch>=1.0.0->bert-score==0.3.13)
  Using cached nvidia_nvtx_cu12-12.1.105-py3-none-manylinux1_x86_64.whl.metadata (1.7 kB)
Requirement already satisfied: triton==2.3.1 in /usr/local/lib/python3.10/dist-packages (from torch>=1.0.0->bert-score==0.3.13)
(2.3.1)
Collecting nvidia-nvjitlink-cu12 (from nvidia-cusolver-cu12==11.4.5.107->torch>=1.0.0->bert-score==0.3.13)
  Using cached nvidia_nvjitlink_cu12-12.6.20-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-packages (from transformers>=3.0.0->bert-scor
e==0.3.13) (2024.5.15)
Requirement already satisfied: safetensors>=0.4.1 in /usr/local/lib/python3.10/dist-packages (from transformers>=3.0.0->bert-sco
re==0.3.13) (0.4.4)
Requirement already satisfied: tokenizers<0.20,>=0.19 in /usr/local/lib/python3.10/dist-packages (from transformers>=3.0.0->bert
-score==0.3.13) (0.19.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->bert-score==0.3.13)
(1.2.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib->bert-score==0.3.13) (0.
12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->bert-score==0.3.1
3) (4.53.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->bert-score==0.3.1
3) (1.4.5)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->bert-score==0.3.13)
(9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->bert-score==0.3.13)
(3.1.2)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets>=2.0.0
->evaluate==0.4.1) (2.3.4)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets>=2.0.0->evalu
ate==0.4.1) (1.3.1)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets>=2.0.0->evaluate
==0.4.1) (24.2.0)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets>=2.0.0->eval
uate==0.4.1) (1.4.1)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets>=2.0.0->ev
aluate==0.4.1) (6.0.5)
```

```
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets>=2.0.0->evaluat
e==0.4.1) (1.9.4)
Requirement already satisfied: async-timeout<5.0,>=4.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets>=2.0.0
->evaluate==0.4.1) (4.0.3)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas->evaluat
e==0.4.1) (1.16.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->torch>=1.0.0->bert-score
==0.3.13) (2.1.5)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from sympy->torch>=1.0.0->bert-sco
re==0.3.13) (1.3.0)
Using cached evaluate-0.4.1-py3-none-any.whl (84 kB)
Using cached bert_score-0.3.13-py3-none-any.whl (61 kB)
Using cached datasets-2.20.0-py3-none-any.whl (547 kB)
Using cached dill-0.3.8-py3-none-any.whl (116 kB)
Using cached fsspec-2024.5.0-py3-none-any.whl (316 kB)
Using cached responses-0.18.0-py3-none-any.whl (38 kB)
Using cached nvidia_cublas_cu12-12.1.3.1-py3-none-manylinux1_x86_64.whl (410.6 MB)
Using cached nvidia_cuda_cupti_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (14.1 MB)
Using cached nvidia_cuda_nvrtc_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (23.7 MB)
Using cached nvidia_cuda_runtime_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (823 kB)
Using cached nvidia_cudnn_cu12-8.9.2.26-py3-none-manylinux1_x86_64.whl (731.7 MB)
Using cached nvidia_cufft_cu12-11.0.2.54-py3-none-manylinux1_x86_64.whl (121.6 MB)
Using cached nvidia_curand_cu12-10.3.2.106-py3-none-manylinux1_x86_64.whl (56.5 MB)
Using cached nvidia_cusolver_cu12-11.4.5.107-py3-none-manylinux1_x86_64.whl (124.2 MB)
Using cached nvidia_cusparse_cu12-12.1.0.106-py3-none-manylinux1_x86_64.whl (196.0 MB)
Using cached nvidia_nccl_cu12-2.20.5-py3-none-manylinux2014_x86_64.whl (176.2 MB)
Using cached nvidia_nvtx_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (99 kB)
Downloading multiprocess-0.70.16-py310-none-any.whl (134 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 134.8/134.8 kB 1.8 MB/s eta 0:00:00
Downloading xxhash-3.4.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (194 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 194.1/194.1 kB 11.9 MB/s eta 0:00:00
Downloading pyarrow-17.0.0-cp310-cp310-manylinux_2_28_x86_64.whl (39.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 39.9/39.9 MB 9.8 MB/s eta 0:00:00
Using cached nvidia_nvjitlink_cu12-12.6.20-py3-none-manylinux2014_x86_64.whl (19.7 MB)
Installing collected packages: xxhash, pyarrow, nvidia-nvtx-cu12, nvidia-nvjitlink-cu12, nvidia-nccl-cu12, nvidia-curand-cu12, n
vidia-cufft-cu12, nvidia-cuda-runtime-cu12, nvidia-cuda-nvrtc-cu12, nvidia-cuda-cupti-cu12, nvidia-cublas-cu12, fsspec, dill, re
sponses, nvidia-cusparse-cu12, nvidia-cudnn-cu12, multiprocess, nvidia-cusolver-cu12, datasets, evaluate, bert-score
  Attempting uninstall: pyarrow
    Found existing installation: pyarrow 14.0.2
    Uninstalling pyarrow-14.0.2:
ERROR: Operation cancelled by user
```

In [8]:
```
! pip install llama_cpp_python[gpu]==0.2.28
```

```
WARNING: Ignoring invalid distribution -yarrow (/usr/local/lib/python3.10/dist-packages)
Requirement already satisfied: llama_cpp_python==0.2.28 in /usr/local/lib/python3.10/dist-packages (from llama_cpp_python[gpu]==
0.2.28) (0.2.28)
Requirement already satisfied: typing-extensions>=4.5.0 in /usr/local/lib/python3.10/dist-packages (from llama_cpp_python==0.2.2
8->llama_cpp_python[gpu]==0.2.28) (4.12.2)
Requirement already satisfied: numpy>=1.20.0 in /usr/local/lib/python3.10/dist-packages (from llama_cpp_python==0.2.28->llama_cp
p_python[gpu]==0.2.28) (1.26.4)
Requirement already satisfied: diskcache>=5.6.1 in /usr/local/lib/python3.10/dist-packages (from llama_cpp_python==0.2.28->llama
_cpp_python[gpu]==0.2.28) (5.6.3)
WARNING: llama-cpp-python 0.2.28 does not provide the extra 'gpu'
WARNING: Ignoring invalid distribution -yarrow (/usr/local/lib/python3.10/dist-packages)
```

In [14]:
```python
import torch

# Check if a GPU is available
if torch.cuda.is_available():
    print("GPU is available and ready!")
    print(f"GPU Name: {torch.cuda.get_device_name(0)}")
else:
    print("GPU is not available.")
```

```
GPU is available and ready!
GPU Name: Tesla T4
```

In [13]:
```python
# Basic Imports for Libraries
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score

import pandas as pd
import numpy as np
from tqdm import tqdm
import json
import re

import torch
#import evaluate

from google.colab import drive
import locale
```

Question 10: Importing Libaries and Setting up Mistral Model (2 Marks)

https://huggingface.co/TheBloke/Mistral-7B-Instruct-v0.2-GGUF/blob/main/mistral-7b-instruct-v0.2.Q5_K_M.gguf

In [12]:
```
## Import Hf_hub_download from hugging_face_hub
## Import Llama from llama_cpp
from huggingface_hub import hf_hub_download
from llama_cpp import Llama
```

In [ ]:
```
! pip install --upgrade llama_cpp_python
```

```
Requirement already satisfied: llama_cpp_python in /usr/local/lib/python3.10/dist-packages (0.2.28)
Collecting llama_cpp_python
  Downloading llama_cpp_python-0.2.87.tar.gz (80.5 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 80.5/80.5 MB 9.9 MB/s eta 0:00:00
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Installing backend dependencies ... done
  Preparing metadata (pyproject.toml) ... done
Requirement already satisfied: typing-extensions>=4.5.0 in /usr/local/lib/python3.10/dist-packages (from llama_cpp_python) (4.1
2.2)
Requirement already satisfied: numpy>=1.20.0 in /usr/local/lib/python3.10/dist-packages (from llama_cpp_python) (1.26.4)
Requirement already satisfied: diskcache>=5.6.1 in /usr/local/lib/python3.10/dist-packages (from llama_cpp_python) (5.6.3)
Requirement already satisfied: jinja2>=2.11.3 in /usr/local/lib/python3.10/dist-packages (from llama_cpp_python) (3.1.4)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2>=2.11.3->llama_cpp_pytho
n) (2.1.5)
Building wheels for collected packages: llama_cpp_python
  Building wheel for llama_cpp_python (pyproject.toml) ... canceled
ERROR: Operation cancelled by user
```

In [15]:
```
# Define the model name or path as a string (You can find this info from hugging face website) Use Mistral

model_name_or_path = "TheBloke/Mistral-7B-Instruct-v0.2-GGUF"

# Define the model basename as a string, indicating it's in the gguf format

model_basename = "mistral-7b-instruct-v0.2.Q5_K_M.gguf" # the model is in gguf format
```

In [ ]:
```
"""import os

# Replace 'your_hf_token' with your actual Hugging Face token
os.environ['HF_TOKEN'] = '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'""
```

In [ ]:
```
"""from huggingface_hub import login
```

```
# This will use the token stored in the environment variable
login(token=os.getenv('HF_TOKEN'))"""
```

```
The token has not been saved to the git credentials helper. Pass `add_to_git_credential=True` in this function directly or `--ad
d-to-git-credential` if using via `huggingface-cli` if you want to set the git credential as well.
Token is valid (permission: fineGrained).
Your token has been saved to /root/.cache/huggingface/token
Login successful
```

In [ ]:
```
# Add Hugging Face token to Colab secrets
#from google.colab import output
#output.eval_js("google.colab.secrets.put('HF_TOKEN', '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')")
```

In [17]:
```
model_path = hf_hub_download(
    repo_id=model_name_or_path,
    filename=model_basename
    )
```

In [18]:
```
# Create an instance of the 'Llama' class with specified parameters
# remove the blank spaces and complete the code

lcpp_llm = Llama(
        model_path=model_path,
        n_threads=2,  # CPU cores
        n_batch=6,  # Should be between 1 and n_ctx, consider the amount of VRAM in your GPU.
        n_gpu_layers=43,  # Change this value based on your model and your GPU VRAM pool.
        n_ctx=4096,  # Context window
    )
```

```
AVX = 1 | AVX_VNNI = 0 | AVX2 = 1 | AVX512 = 0 | AVX512_VBMI = 0 | AVX512_VNNI = 0 | FMA = 1 | NEON = 0 | ARM_FMA = 0 | F16C = 1
| FP16_VA = 0 | WASM_SIMD = 0 | BLAS = 0 | SSE3 = 1 | SSSE3 = 1 | VSX = 0 |
```

Section 3: Text to Label (12 Marks)

Zero-Shot Prompting (6 Marks)

Q11: Define the Prompt Template, System Message, generate_prompt (3 Marks)

Define a **system message** as a string and assign it to the variable system_message to generate product class.

Create a **zero shot prompt template** that incorporates the system message and user input.

Define **generate_prompt** function that takes both the system_message and user_input as arguments and formats them into a prompt template

Write a Python function called **generate_mistral_response** that takes a single parameter, narrative, which represents the user's complain. Inside the function, you should perform the following tasks:

- **Combine the system_message and narrative to create a prompt string using generate_prompt function.**

Generate a response from the Mistral model using the lcpp_llm instance with the following parameters:

- prompt should be the combined prompt string.
- max_tokens should be set to 1200.
- temperature should be set to 0.
- top_p should be set to 0.95.
- repeat_penalty should be set to 1.2.
- top_k should be set to 50.
- stop should be set as a list containing '/s'.
- echo should be set to False. Extract and return the response text from the generated response.

Don't forget to provide a value for the system_message variable before using it in the function.

In [4]:
```
system_message = """You are a product classification assistant. Given a narrative, classify the product category"""
```

In [5]:
```
zero_shot_prompt_template = """You are a product classification assistant. Given a complaint narrative, classify the product rela

Complaint Narrative: {user_input}

Product Classification:"""
```

In [6]:
```
# Define function that combines user_prompt and system_message to create the prompt
def generate_prompt(system_message,user_input):
    prompt = f"{system_message}\n\nClassify the following complaint:\n\n{user_input}\n\nCategory:"
    return prompt
```

In [7]:
```
def generate_mistral_response(input_text):
```

```
    # Combine user_prompt and system_message to create the prompt
    prompt = generate_prompt(system_message,input_text)

    # Generate a response from the LLaMA model
    response = lcpp_llm(
        prompt=prompt,
        max_tokens=1200,
        temperature=0,
        top_p=0.95,
        repeat_penalty=1.2,
        top_k=50,
        stop=['/s'],
        echo=False
    )

    # Extract and return the response text
    response_text = response["choices"][0]['text']  ### Fill in the blank
    return response_text
```

Due to limited GPU resources, we will test our model with zero prompts on only 50 examples instead of the entire dataset.

In [8]:
```
# Randomly select 50 rows
new_data = data.sample(n=5, random_state=40)
```

In [9]:
```
new_data
```

Out[9]:

	product	narrative	summary
167	retail_banking	fraudulent charge totaling made capital one ch...	A fraudulent charge was made on the individual...
169	credit_reporting	block except otherwise provided section consum...	The text outlines various stipulations regardi...
461	credit_card	usaa master plan collect cancellation debt usa...	The input appears to be a complaint about USAA...
253	credit_reporting	block except otherwise provided section consum...	The text pertains to the stipulations and oper...
42	credit_reporting	open account acct opened balance account acct ...	The input is about various accounts being open...

Q12: Create a new column in the DataFrame called 'mistral_response' and populate it with responses generated by applying the 'generate_mistral_response' function to each 'narrative' in

the DataFrame and prepare the mistral_response_cleaned column using extract_category function (1 Marks)

In [22]:
```
!pip show llama_cpp_python
```

```
WARNING: Ignoring invalid distribution -yarrow (/usr/local/lib/python3.10/dist-packages)
Name: llama_cpp_python
Version: 0.2.28
Summary: Python bindings for the llama.cpp library
Home-page: https://github.com/abetlen/llama-cpp-python
Author:
Author-email: Andrei Betlen <abetlen@gmail.com>
License: MIT
Location: /usr/local/lib/python3.10/dist-packages
Requires: diskcache, numpy, typing-extensions
Required-by:
```

In [23]:
```
len(new_data)
```

Out[23]:
```
5
```

In [2]:
```
"""import time
import subprocess

def monitor_gpu_usage(interval=5):
         "Monitor GPU usage every 'interval' seconds."
while True:
        # Run the nvidia-smi command and print its output
     gpu_usage = subprocess.run(['nvidia-smi'], stdout=subprocess.PIPE).stdout.decode('utf-8')
     print(gpu_usage)
time.sleep(interval)

# Start monitoring in a separate thread
import threading
monitor_thread = threading.Thread(target=monitor_gpu_usage, daemon=True)
monitor_thread.start()
# example - new_data['mistral_response'] = new_data['narrative'].apply(lambda x:_____ )
new_data['mistral_response'] = new_data['narrative'].apply(lambda x: generate_mistral_response(x))

# Stop the monitoring once done (optional, since daemon thread will stop with the script)
monitor_thread.join(timeout=1)
"""
```

Out[2]:
```
'import time\nimport subprocess\n\ndef monitor_gpu_usage(interval=5):\n         "Monitor GPU usage every \'interval\' second
s."\nwhile True:\n         # Run the nvidia-smi command and print its output\n     gpu_usage = subprocess.run([\'nvidia-smi\'], s
tdout=subprocess.PIPE).stdout.decode(\'utf-8\')\n     print(gpu_usage)\ntime.sleep(interval)\n\n# Start monitoring in a separate
thread\nimport threading\nmonitor_thread = threading.Thread(target=monitor_gpu_usage, daemon=True)\nmonitor_thread.start()\n# ex
ample - new_data[\'mistral_response\'] = new_data[\'narrative\'].apply(lambda x:_____ )\nnew_data[\'mistral_response\'] = new_d
ata[\'narrative\'].apply(lambda x: generate_mistral_response(x))\n\n# Stop the monitoring once done (optional, since daemon thre
ad will stop with the script)\nmonitor_thread.join(timeout=1)\n'
```

In [31]:
```
# example - new_data['mistral_response'] = new_data['narrative'].apply(lambda x:_____ )
new_data['mistral_response'] = new_data['narrative'].apply(lambda x: generate_mistral_response(x))
```

```
Llama.generate: prefix-match hit
Llama.generate: prefix-match hit
Llama.generate: prefix-match hit
Llama.generate: prefix-match hit
Llama.generate: prefix-match hit
```

In [ ]:
```
#new_data['mistral_response'] = # Prepare the 'mistral_response_cleaned' column using the extract_category function
```

In [39]:
```
def extract_category(text):
    # Define the regex pattern to match "category:" or "Category:" followed by a word
    pattern = r'category:\s*(\w+)'  # The pattern itself remains the same

    # Use re.search with the re.IGNORECASE flag to make it case-insensitive
    match = re.search(pattern, text, re.IGNORECASE)

    # If a match is found, return the captured group, else return None
    if match:
        return match.group(1)
    else:
        pattern1 = r'(credit_card|retail_banking|credit_reporting|mortgages_and_loans|debt_collection)'
        match = re.search(pattern1, text, re.IGNORECASE)
        if match:
            return match.group()
        else:
            return ''
```

In [50]:
```
# example - new_data['mistral_response_cleaned'] = new_data['narrative'].apply(lambda x:_____ )
new_data['mistral_response_cleaned'] = new_data['narrative'].apply(lambda x: extract_category(x))
```

Q14: Calculate the F1 score (1 Marks)

In [49]:
```
new_data.head()
```

Out[49]:

	product	narrative	summary	mistral_response	mistral_response_cleaned
167	retail_banking	fraudulent charge totaling made capital one ch...	A fraudulent charge was made on the individual...	Financial Services / Banking - Fraud or Secur...	
169	credit_reporting	block except otherwise provided section consum...	The text outlines various stipulations regardi...	Consumer Protection/Identity Theft. This comp...	
461	credit_card	usaa master plan collect cancellation debt usa...	The input appears to be a complaint about USAA...	Identity Theft/Fraud.\n\nExplanation: Based o...	
253	credit_reporting	block except otherwise provided section consum...	The text pertains to the stipulations and oper...	Consumer Protection/Identity Theft. This comp...	
42	credit_reporting	open account acct opened balance account acct ...	The input is about various accounts being open...	Banking Services (specifically, checking or s...	

In [51]:
```
from sklearn.metrics import f1_score
# Calculate F1 score for 'product' and 'mistral_response'

# Calculate F1 score for 'product' (actual values) and 'mistral_response_cleaned' (predicted values)
f1 =  f1_score(new_data['product'], new_data['mistral_response'], average= 'macro')

print(f'F1 Score: {f1}')
```

```
F1 Score: 0.0
```

In [52]:
```
# Calculate F1 score for 'product' and 'mistral_response_cleaned'

# Calculate F1 score for 'product' (actual values) and 'mistral_response_cleaned' (predicted values)
f2 =  f1_score(new_data['product'], new_data['mistral_response_cleaned'], average='macro')
print(f'F2 Score: {f2}')
```

```
F2 Score: 0.0
```

In [42]:
```
new_data.head()
```

Out[42]:

	product	narrative	summary	mistral_response	mistral_response_cleaned
167	retail_banking	fraudulent charge totaling made capital one ch...	A fraudulent charge was made on the individual...	Financial Services / Banking - Fraud or Secur...	
169	credit_reporting	block except otherwise provided section consum...	The text outlines various stipulations regardi...	Consumer Protection/Identity Theft. This comp...	
461	credit_card	usaa master plan collect cancellation debt usa...	The input appears to be a complaint about USAA...	Identity Theft/Fraud.\n\nExplanation: Based o...	
253	credit_reporting	block except otherwise provided section consum...	The text pertains to the stipulations and oper...	Consumer Protection/Identity Theft. This comp...	
42	credit_reporting	open account acct opened balance account acct ...	The input is about various accounts being open...	Banking Services (specifically, checking or s...	

Q15: Explain the difference in F1 scores between mistral_response and mistral_response_cleaned. (1 Marks)

The F1 score measures the balance between precision and recall for a classification model, with a focus on the ability to correctly identify positive instances.

Difference Between mistral_response and mistral_response_cleaned F1 Scores: mistral_response F1 Score:

Description: This F1 score is computed using the raw responses generated by the Mistral model. These responses may include additional information, irrelevant details, or might not follow a structured format. Expected Value: The F1 score for mistral_response may be lower if the responses are not consistently formatted or contain extraneous information that makes it harder to match with the actual product categories. mistral_response_cleaned F1 Score:

Description: This F1 score is calculated using the cleaned responses, which are processed to extract specific categories using the extract_category function. This function uses regex patterns to focus on relevant category information, potentially filtering out irrelevant details. Expected Value: The F1 score for mistral_response_cleaned is likely to be higher if the cleaning process effectively extracts meaningful categories from the raw responses, thus improving the accuracy of the predicted categories. Summary: The difference in F1 scores indicates how well the cleaning and extraction process improves the quality of the predicted categories:

Higher F1 Score for mistral_response_cleaned: Suggests that the cleaning process is effective in improving the relevance and accuracy of the extracted categories. Lower F1 Score for mistral_response: Indicates that raw responses may not be as useful or accurate without additional

processing. In summary, the F1 score difference highlights the impact of data preprocessing on the performance of the classification task.

Few-Shot Prompting (6 Marks)

Q16: Prepare examples for a few-shot prompt, formulate the prompt, and generate the Mistral response. (4 Marks)

Generate a set of examples by randomly selecting 10 instances of user_input and assistant_output from dataset ensuring a balanced representation with 2 examples from each class.

In [53]:
```
# Separate categories
import json
review_1 = data[data['product'] == 'credit_card']
review_2 = data[data['product'] == 'retail_banking']
review_3 = data[data['product'] == 'credit_reporting']
review_4 = data[data['product'] == 'mortgages_and_loans']
review_5 = data[data['product'] == 'debt_collection']

# Sample 2 examples for each category
examples_1 = review_1.sample(2, random_state=40)
examples_2 = review_2.sample(2, random_state=40)
examples_3 = review_3.sample(2, random_state=40)
examples_4 = review_4.sample(2, random_state=40)
examples_5 = review_5.sample(2, random_state=40)

# Concatenate examples for few shot prompting
examples_df = pd.concat([examples_1,examples_2,examples_3,examples_4,examples_5 ])

# Create the gold examples for evaluation set by excluding examples
gold_examples_df = data.drop(index=examples_df.index)

# Convert examples to JSON
columns_to_select = ['narrative', 'product']
examples_json = examples_df[columns_to_select].to_json(orient='records')

# Print the first record from the JSON
print(json.loads(examples_json)[0])

# Print the shapes of the datasets
```

```
print("Examples Set Shape:", examples_df.shape)
print("Gold Examples Shape:", gold_examples_df.shape)
```

```
{'narrative': 'called request new york state covid relief plan day interest fee waived amex provided relief leading late payment
amex refused honor relief day gap insists charging late fee', 'product': 'credit_card'}
Examples Set Shape: (10, 3)
Gold Examples Shape: (490, 3)
```

Define your **system_message**.

Define **first_turn_template**, **example_template** and **prediction template**

create few shot prompt using gold examples and system_message

Randomly select 50 rows from test_df as test_data

Create **mistral_response** and **mistral_response_cleaned** columns

In [54]:
```
system_message = """You are an AI assistant trained to classify customer complaints into specific product categories based on the
```

In [55]:
```
mistral_first_turn_template = """<s>[INST]\n <<SYS>> \n {system_message} \n <</SYS>>```{user_message}``` /n [/INST] \n{assistant_
mistral_examples_template =  """<s>[INST]\n ```{user_message}``` \n [/INST] \n {assistant_message}\n</s>"""
prediction_template = """<s>[INST]\n ```{user_message}```[/INST]"""
```

In [ ]:
```

```

In [56]:
```
def create_few_shot_prompt(system_message, examples):

    """
    Return a prompt message in the format expected by Mistral 7b.
    10 examples are selected randomly as golden examples to form the
    few-shot prompt.
    We then loop through each example and parse the narrative as the user message
    and the product as the assistant message.

    Args:
        system_message (str): system message with instructions for classification
        examples(DataFrame): A DataFrame with examples (product + narrative + summary)
        to form the few-shot prompt.

    Output:
        few_shot_prompt (str): A prompt string in the Mistral format
```

```
    """

    few_shot_prompt = f"{system_message}\n\n"

    columns_to_select = examples_df[['narrative', 'product']].columns.tolist()

    examples = (
        examples_df.loc[:, columns_to_select].to_json(orient='records')
    )

    for idx, example in enumerate(json.loads(examples)):
        user_input_example = example['narrative']
        assistant_output_example = example['product']

        if idx == 0:
            few_shot_prompt += mistral_first_turn_template.format(
                system_message=system_message,
                user_message=user_input_example,
                assistant_message=assistant_output_example
            )
        else:
            few_shot_prompt += mistral_examples_template.format(
                user_message=user_input_example,
                assistant_message=assistant_output_example
            )

    return few_shot_prompt
```

In [ ]:

In [57]:
```
few_shot_prompt = create_few_shot_prompt(system_message, examples_df)
```

In [58]:
```
def generate_prompt(few_shot_prompt,new_review):
    prompt = f"{few_shot_prompt}\n\nComplaint: {new_review}\n\nBased on the examples provided, classify this complaint into one c
    return prompt
```

In [59]:
```
def generate_mistral_response(input_text):

    # Combine user_prompt and system_message to create the prompt
    prompt =  generate_prompt(few_shot_prompt, input_text)

    # Generate a response from the LLaMA model
```

```
    response = lcpp_llm(
        prompt=prompt,
        max_tokens=1200,
        temperature=0,
        top_p=0.95,
        repeat_penalty=1.2,
        top_k=50,
        stop=['/s'],
        echo=False

    )

    # Extract and return the response text
    response_text = response["choices"][0]['text']  ### Fill in the blank
    return response_text
```

In [60]:
```
# Randomly select 50 rows
new_data = gold_examples_df.sample(n=5, random_state=40)
```

In [63]:
```
new_data['mistral_response'] = new_data['narrative'].apply(lambda x: generate_mistral_response(x))
```

```
Llama.generate: prefix-match hit
Llama.generate: prefix-match hit
Llama.generate: prefix-match hit
Llama.generate: prefix-match hit
Llama.generate: prefix-match hit
```

In [64]:
```
new_data.head()
```

Out[64]:

	product	narrative	summary	mistral_response
74	credit_reporting	anyone us credit report another consumer repor...	The narrator is filing a complaint against ind...	The given text appears to be discussing a dis...
174	credit_reporting	block except otherwise provided section consum...	The input refers to regulations for consumer r...	The given text appears to be discussing an is...
85	credit_reporting	shocked reviewed credit report found late paym...	The user reviewed their credit report and was ...	\n\ncredit_card\n\nExplanation: The text ment...
372	credit_reporting	except otherwise provided section consumer rep...	This passage discusses the procedures and requ...	The given text appears to be a customer's acc...
296	credit_reporting	block except otherwise provided section consum...	The input text discusses the obligations of co...	The given text appears to be discussing an is...

In [65]:
```
new_data['mistral_response_cleaned'] = new_data['mistral_response'].apply(lambda x: extract_category(x))
```

In [67]:
```
new_data.head()
```

Out[67]:

	product	narrative	summary	mistral_response	mistral_response_cleaned
74	credit_reporting	anyone us credit report another consumer repor...	The narrator is filing a complaint against ind...	The given text appears to be discussing a dis...	
174	credit_reporting	block except otherwise provided section consum...	The input refers to regulations for consumer r...	The given text appears to be discussing an is...	
85	credit_reporting	shocked reviewed credit report found late paym...	The user reviewed their credit report and was ...	\n\ncredit_card\n\nExplanation: The text ment...	
372	credit_reporting	except otherwise provided section consumer rep...	This passage discusses the procedures and requ...	The given text appears to be a customer's acc...	
296	credit_reporting	block except otherwise provided section consum...	The input text discusses the obligations of co...	The given text appears to be discussing an is...	

Calculate F1 score (1 Mark)

In [66]:
```
# Calculate F1 score for 'product' and 'mistral_response_cleaned'
f3 =  f1_score(new_data['product'], new_data['mistral_response_cleaned'], average='macro')
```

```
print(f'F1 Score: {f3}')
```

```
F1 Score: 0.0
```

Q17: Share your observations on the few-shot and zero-shot prompt techniques. (1 Marks)

Few-Shot Prompting:

Performance:

Effectiveness: Few-shot prompting typically provides better results than zero-shot prompting because it uses specific examples to guide the model's responses. By including examples in the prompt, the model can better understand the context and expected output. F1 Score: In practice, few-shot prompting often leads to higher F1 scores, as the model leverages the provided examples to make more accurate predictions. Customization:

Tailored Outputs: It allows for more customization by including examples relevant to the task. This helps the model adapt to specific nuances in the data or task. Contextual Understanding: With examples, the model can better grasp the intricacies of the desired classification or response. Implementation Complexity:

Data Preparation: Requires careful selection and formatting of examples to ensure they are representative of the task. This can involve additional preprocessing and data handling. Zero-Shot Prompting:

Performance:

Baseline Performance: Zero-shot prompting provides a baseline performance without any task-specific examples. The model makes predictions based on its pre-trained knowledge and general understanding. F1 Score: Generally, zero-shot performance might be lower compared to few-shot due to the lack of specific examples guiding the model. Flexibility:

Simplicity: No need for example-specific data preparation. This makes zero-shot prompting simpler and quicker to implement, as it relies on the model’s general capabilities. Use Case:

General Applicability: Suitable for tasks where examples are not readily available or where rapid deployment is needed. It leverages the model's broad training data to infer responses. Comparison:

Accuracy: Few-shot prompting often results in higher accuracy and better performance metrics, as the model benefits from specific examples that guide its responses. Resource Requirements: Few-shot prompting may require more effort in selecting and formatting examples but provides

more targeted results. Zero-shot prompting is quicker to implement but may not achieve the same level of precision. Conclusion:

Few-shot prompting is generally preferred when high accuracy and task-specific guidance are required, whereas zero-shot prompting is useful for simpler applications or when examples are not available. Balancing the two techniques depends on the specific needs of the task and available resources.

Section 4: Text to Text generation (5 Marks)

Define a **system message** as a string and assign it to the variable system_message to generate product class. **(1 Mark)**

Create a **zero shot prompt template** that incorporates the system message and user input.

Define **generate_prompt** function that takes both the system_message and user_input as arguments and formats them into a prompt template

Write a Python function called **generate_mistral_response** that takes a single parameter, narrative, which represents the user's complain. Inside the function, you should perform the following tasks:

- **Combine the system_message and narrative to create a prompt string using generate_prompt function.**

Generate a response from the Mistral model using the lcpp_llm instance with the following parameters:

- prompt should be the combined prompt string.
- max_tokens should be set to 1200.
- temperature should be set to 0.
- top_p should be set to 0.95.
- repeat_penalty should be set to 1.2.
- top_k should be set to 50.
- stop should be set as a list containing '/s'.
- echo should be set to False. Extract and return the response text from the generated response.

Don't forget to provide a value for the system_message variable before using it in the function.

In [68]:
```
system_message = """You are a classification assistant that categorizes customer complaints into one of the following product cat
```

In [69]:
```
zero_shot_prompt_template = """You are a classification assistant that categorizes customer complaints into one of the following

Complaint Narrative: {user_input}

Product Classification:"""
```

In [70]:
```
# Define function that combines user_prompt and system_message to create the prompt
def generate_prompt(system_message,user_input):
    prompt = zero_shot_prompt_template.format(
        system_message=system_message,
        user_input=user_input
    )
    return prompt
```

In [71]:
```
def generate_mistral_response(input_text):

    # Combine user_prompt and system_message to create the prompt
    prompt = generate_prompt(system_message, input_text)

    # Generate a response from the LLaMA model
    response = lcpp_llm(
         prompt=prompt,
        max_tokens=1200,
        temperature=0,
        top_p=0.95,
        repeat_penalty=1.2,
        top_k=50,
        stop=['/s'],
        echo=False

    )

    # Extract and return the response text
    response_text = response["choices"][0]["text"]  ### Fill in the blank
    return response_text
```

Q19: Generate mistral_response column containing LLM generated summaries (1 Marks)

In [72]:
```
# Randomly select 50 rows
gold_examples = data.sample(n=5, random_state=40)
```

In [73]:
```
data.head()
```

Out[73]:

	product	narrative	summary
0	credit_card	purchase order day shipping amount receive pro...	The customer made a purchase order with an agr...
1	credit_card	forwarded message date tue subject please inve...	The sender of the email believes they have bee...
2	retail_banking	forwarded message cc sent friday pdt subject f...	The sender of the email alleges that Wells Far...
3	credit_reporting	payment history missing credit report speciali...	The credit report from Specialized Loan Servic...
4	credit_reporting	payment history missing credit report made mis...	The text concerns a person who found an unauth...

In [87]:
```
gold_examples['mistral_response'] = gold_examples['narrative'].apply(lambda x: generate_mistral_response(x))
```

```
Llama.generate: prefix-match hit
Llama.generate: prefix-match hit
Llama.generate: prefix-match hit
Llama.generate: prefix-match hit
Llama.generate: prefix-match hit
```

In [88]:
```
gold_examples.head()
```

Out[88]:

	product	narrative	summary	mistral_response
461	credit_card	usaa master plan collect cancellation debt usa...	The input appears to be a complaint about USAA...	credit_card. This complaint is about unauthor...
167	retail_banking	fraudulent charge totaling made capital one ch...	A fraudulent charge was made on the individual...	credit_card. This complaint is about a fraudu...
169	credit_reporting	block except otherwise provided section consum...	The text outlines various stipulations regardi...	credit_reporting. This complaint is about a d...
42	credit_reporting	open account acct opened balance account acct ...	The input is about various accounts being open...	retail_banking\n\nExplanation: The complaint ...
253	credit_reporting	block except otherwise provided section consum...	The text pertains to the stipulations and oper...	credit_reporting. This complaint is about a d...

Q20: Evaluate bert score (2 Marks)

In [75]:
```
def evaluate_score(result, scorer, bert_score=False):

    """
    Return the ROUGE score or BERTScore for predictions on gold examples
    For each example we make a prediction using the prompt.
    Gold summaries and the AI generated summaries are aggregated into lists.
    These lists are used by the corresponding scorers to compute metrics.
    Since BERTScore is computed for each candidate-reference pair, we take the
    average F1 score across the gold examples.

    Args:
        prompt (List): list of messages in the Open AI prompt format
        gold_examples (str): JSON string with list of gold examples
        scorer (function): Scorer function used to compute the ROUGE score or the
                           BERTScore
        bert_score (boolean): A flag variable that indicates if BERTScore should
                              be used as the metric.

    Output:
        score (float): BERTScore or ROUGE score computed by comparing model predictions
                       with ground truth
    """

    model_predictions = result['mistral_response'].tolist
    ground_truths = result['summary'].tolist()
    if bert_score:
        score = scorer.compute(
            predictions=model_predictions,
            references=ground_truths,
            lang="en",
            rescale_with_baseline=True
        )

        return sum(score['f1'])/len(score['f1'])
    else:
        return scorer.compute(
            predictions=model_predictions,
            references=ground_truths
        )
```

In [79]:
```
! pip install bert-score
```

```
WARNING: Ignoring invalid distribution -yarrow (/usr/local/lib/python3.10/dist-packages)
Collecting bert-score
  Using cached bert_score-0.3.13-py3-none-any.whl.metadata (15 kB)
Requirement already satisfied: torch>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from bert-score) (2.3.1+cu121)
Requirement already satisfied: pandas>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from bert-score) (2.1.4)
Requirement already satisfied: transformers>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from bert-score) (4.42.4)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from bert-score) (1.26.4)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from bert-score) (2.32.3)
Requirement already satisfied: tqdm>=4.31.1 in /usr/local/lib/python3.10/dist-packages (from bert-score) (4.66.5)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from bert-score) (3.7.1)
Requirement already satisfied: packaging>=20.9 in /usr/local/lib/python3.10/dist-packages (from bert-score) (24.1)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.1->bert-scor
e) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.1->bert-score) (2024.1)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.1->bert-score) (2024.
1)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch>=1.0.0->bert-score) (3.15.4)
Requirement already satisfied: typing-extensions>=4.8.0 in /usr/local/lib/python3.10/dist-packages (from torch>=1.0.0->bert-scor
e) (4.12.2)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch>=1.0.0->bert-score) (1.13.1)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch>=1.0.0->bert-score) (3.3)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch>=1.0.0->bert-score) (3.1.4)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from torch>=1.0.0->bert-score) (2024.6.1)
Collecting nvidia-cuda-nvrtc-cu12==12.1.105 (from torch>=1.0.0->bert-score)
  Using cached nvidia_cuda_nvrtc_cu12-12.1.105-py3-none-manylinux1_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-runtime-cu12==12.1.105 (from torch>=1.0.0->bert-score)
  Using cached nvidia_cuda_runtime_cu12-12.1.105-py3-none-manylinux1_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-cupti-cu12==12.1.105 (from torch>=1.0.0->bert-score)
  Using cached nvidia_cuda_cupti_cu12-12.1.105-py3-none-manylinux1_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cudnn-cu12==8.9.2.26 (from torch>=1.0.0->bert-score)
  Using cached nvidia_cudnn_cu12-8.9.2.26-py3-none-manylinux1_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cublas-cu12==12.1.3.1 (from torch>=1.0.0->bert-score)
  Using cached nvidia_cublas_cu12-12.1.3.1-py3-none-manylinux1_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cufft-cu12==11.0.2.54 (from torch>=1.0.0->bert-score)
  Using cached nvidia_cufft_cu12-11.0.2.54-py3-none-manylinux1_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-curand-cu12==10.3.2.106 (from torch>=1.0.0->bert-score)
  Using cached nvidia_curand_cu12-10.3.2.106-py3-none-manylinux1_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cusolver-cu12==11.4.5.107 (from torch>=1.0.0->bert-score)
  Using cached nvidia_cusolver_cu12-11.4.5.107-py3-none-manylinux1_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cusparse-cu12==12.1.0.106 (from torch>=1.0.0->bert-score)
  Using cached nvidia_cusparse_cu12-12.1.0.106-py3-none-manylinux1_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-nccl-cu12==2.20.5 (from torch>=1.0.0->bert-score)
  Using cached nvidia_nccl_cu12-2.20.5-py3-none-manylinux2014_x86_64.whl.metadata (1.8 kB)
Collecting nvidia-nvtx-cu12==12.1.105 (from torch>=1.0.0->bert-score)
```

```
  Using cached nvidia_nvtx_cu12-12.1.105-py3-none-manylinux1_x86_64.whl.metadata (1.7 kB)
Requirement already satisfied: triton==2.3.1 in /usr/local/lib/python3.10/dist-packages (from torch>=1.0.0->bert-score) (2.3.1)
Collecting nvidia-nvjitlink-cu12 (from nvidia-cusolver-cu12==11.4.5.107->torch>=1.0.0->bert-score)
  Using cached nvidia_nvjitlink_cu12-12.6.20-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Requirement already satisfied: huggingface-hub<1.0,>=0.23.2 in /usr/local/lib/python3.10/dist-packages (from transformers>=3.0.0
->bert-score) (0.23.2)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from transformers>=3.0.0->bert-score) (6.
0.2)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-packages (from transformers>=3.0.0->bert-scor
e) (2024.5.15)
Requirement already satisfied: safetensors>=0.4.1 in /usr/local/lib/python3.10/dist-packages (from transformers>=3.0.0->bert-sco
re) (0.4.4)
Requirement already satisfied: tokenizers<0.20,>=0.19 in /usr/local/lib/python3.10/dist-packages (from transformers>=3.0.0->bert
-score) (0.19.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->bert-score) (1.2.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib->bert-score) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->bert-score) (4.53.
1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->bert-score) (1.4.
5)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->bert-score) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->bert-score) (3.1.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->bert-score)
(3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->bert-score) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->bert-score) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->bert-score) (2024.
7.4)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas>=1.0.1->
bert-score) (1.16.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->torch>=1.0.0->bert-scor
e) (2.1.5)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from sympy->torch>=1.0.0->bert-sco
re) (1.3.0)
Using cached bert_score-0.3.13-py3-none-any.whl (61 kB)
Using cached nvidia_cublas_cu12-12.1.3.1-py3-none-manylinux1_x86_64.whl (410.6 MB)
Using cached nvidia_cuda_cupti_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (14.1 MB)
Using cached nvidia_cuda_nvrtc_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (23.7 MB)
Using cached nvidia_cuda_runtime_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (823 kB)
Using cached nvidia_cudnn_cu12-8.9.2.26-py3-none-manylinux1_x86_64.whl (731.7 MB)
Using cached nvidia_cufft_cu12-11.0.2.54-py3-none-manylinux1_x86_64.whl (121.6 MB)
Using cached nvidia_curand_cu12-10.3.2.106-py3-none-manylinux1_x86_64.whl (56.5 MB)
Using cached nvidia_cusolver_cu12-11.4.5.107-py3-none-manylinux1_x86_64.whl (124.2 MB)
Using cached nvidia_cusparse_cu12-12.1.0.106-py3-none-manylinux1_x86_64.whl (196.0 MB)
```

```
Using cached nvidia_nccl_cu12-2.20.5-py3-none-manylinux2014_x86_64.whl (176.2 MB)
Using cached nvidia_nvtx_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (99 kB)
Using cached nvidia_nvjitlink_cu12-12.6.20-py3-none-manylinux2014_x86_64.whl (19.7 MB)
WARNING: Ignoring invalid distribution -yarrow (/usr/local/lib/python3.10/dist-packages)
Installing collected packages: nvidia-nvtx-cu12, nvidia-nvjitlink-cu12, nvidia-nccl-cu12, nvidia-curand-cu12, nvidia-cufft-cu12,
nvidia-cuda-runtime-cu12, nvidia-cuda-nvrtc-cu12, nvidia-cuda-cupti-cu12, nvidia-cublas-cu12, nvidia-cusparse-cu12, nvidia-cudnn
-cu12, nvidia-cusolver-cu12, bert-score
Successfully installed bert-score-0.3.13 nvidia-cublas-cu12-12.1.3.1 nvidia-cuda-cupti-cu12-12.1.105 nvidia-cuda-nvrtc-cu12-12.
1.105 nvidia-cuda-runtime-cu12-12.1.105 nvidia-cudnn-cu12-8.9.2.26 nvidia-cufft-cu12-11.0.2.54 nvidia-curand-cu12-10.3.2.106 nvi
dia-cusolver-cu12-11.4.5.107 nvidia-cusparse-cu12-12.1.0.106 nvidia-nccl-cu12-2.20.5 nvidia-nvjitlink-cu12-12.6.20 nvidia-nvtx-c
u12-12.1.105
```

In [80]:
```python
from bert_score import BERTScorer

bert_scorer = BERTScorer(lang="en", rescale_with_baseline=True)
```

```
tokenizer_config.json:   0%|          | 0.00/25.0 [00:00<?, ?B/s]
config.json:   0%|          | 0.00/482 [00:00<?, ?B/s]
vocab.json:   0%|          | 0.00/899k [00:00<?, ?B/s]
merges.txt:   0%|          | 0.00/456k [00:00<?, ?B/s]
tokenizer.json:   0%|          | 0.00/1.36M [00:00<?, ?B/s]
model.safetensors:   0%|          | 0.00/1.42G [00:00<?, ?B/s]
Some weights of RobertaModel were not initialized from the model checkpoint at roberta-large and are newly initialized: ['robert
a.pooler.dense.bias', 'roberta.pooler.dense.weight']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
```

In [ ]:
```python
```

In [84]:
```python
# Randomly select 50 rows
#gold_examples = gold_examples.sample(n=5, random_state=40)
```

In [89]:
```python
gold_examples.head()
```

Out[89]:

	product	narrative	summary	mistral_response
461	credit_card	usaa master plan collect cancellation debt usa...	The input appears to be a complaint about USAA...	credit_card. This complaint is about unauthor...
167	retail_banking	fraudulent charge totaling made capital one ch...	A fraudulent charge was made on the individual...	credit_card. This complaint is about a fraudu...
169	credit_reporting	block except otherwise provided section consum...	The text outlines various stipulations regardi...	credit_reporting. This complaint is about a d...
42	credit_reporting	open account acct opened balance account acct ...	The input is about various accounts being open...	retail_banking\n\nExplanation: The complaint ...
253	credit_reporting	block except otherwise provided section consum...	The text pertains to the stipulations and oper...	credit_reporting. This complaint is about a d...

In [91]:
```
#score = evaluate_score(gold_examples, bert_scorer, bert_score=True)
#print(f'BERTScore: {score}')
```

In [108…
```
reference=gold_examples['product'].tolist()
```

In [109…
```
reference
```

Out[109]:
```
['credit_card',
 'retail_banking',
 'credit_reporting',
 'credit_reporting',
 'credit_reporting']
```

In [110…
```
predictions=gold_examples['mistral_response'].tolist()
```

In [111…
```
predictions
```

Out[111]:
```
[" credit_card. This complaint is about unauthorized transactions on a USAA credit card and the bank's response to collecting de
bt from those fraudulent accounts.",
 ' credit_card. This complaint is about a fraudulent charge on the customer\'s Capital One credit card account, as indicated by
several mentions of "fraudulent charge," "disputed charge," and "capital one credit card." The customer also states that they ne
ver lost possession of their debit card but someone was able to make unauthorized purchases with it. This is a clear indication
of credit card fraud.',
 " credit_reporting. This complaint is about a dispute regarding the reporting of information in a consumer's credit report, whi
ch falls under the product category of credit_reporting. The text mentions various sections and subsections from the Fair Credit
Reporting Act (FCRA), specifically those related to identity theft and disputes over reported information. Therefore, it is most
likely that this complaint pertains to an issue with a credit reporting agency or bureau.",
 ' retail_banking\n\nExplanation: The complaint narrative mentions the opening and closing of multiple bank accounts, which is a
common issue in retail banking. Credit cards and mortgages/loans also involve opening accounts but they are typically not mentio
ned as frequently or in such large numbers as in this case. Additionally, credit reporting issues would usually include specific
mention of errors on credit reports rather than just account balances. Therefore, based on the information provided, it is most
likely that this complaint pertains to retail banking.',
 " credit_reporting. This complaint is about a dispute regarding the reporting of information in a consumer's credit report, whi
ch falls under the product category of credit_reporting. The text mentions various sections and subsections from the Fair Credit
Reporting Act (FCRA), specifically those related to identity theft and disputes over reported information. Therefore, it is most
likely that this complaint pertains to an issue with a credit reporting agency or bureau."]
```

In [112…
```
P, R, F1 = bert_scorer.score(predictions, reference)
```

In [113…
```
for i, (p, r, f1) in enumerate(zip(P, R, F1)):
    print(f"Example {i+1}:")
    print(f"  Precision: {p.item():.4f}")
    print(f"  Recall: {r.item():.4f}")
    print(f"  F1: {f1.item():.4f}")
```

```
Example 1:
  Precision: 0.0064
  Recall: 0.6289
  F1: 0.3003
Example 2:
  Precision: -0.2660
  Recall: 0.0497
  F1: -0.1115
Example 3:
  Precision: -0.2535
  Recall: 0.3062
  F1: 0.0122
Example 4:
  Precision: -0.2881
  Recall: 0.2174
  F1: -0.0467
Example 5:
  Precision: -0.2535
  Recall: 0.3062
  F1: 0.0122
```

Q21: Write your observation (1 Marks)

since due to limitation of speed of gpu and technical issues i have taken only 5 gold examples BERT score can be improve with better number of gold examples and prompt

BERTScore Evaluation provides a nuanced measure of text similarity by leveraging contextual embeddings from BERT. It evaluates how well the generated responses align with the ground truth using semantic understanding rather than simple token matching. This makes BERTScore particularly effective for assessing quality in tasks like summarization and translation, where capturing meaning is crucial.

In the context of evaluating the Mistral model's output:

Precision of Evaluation: BERTScore's ability to use contextual embeddings ensures that it captures the subtleties of language, leading to a more accurate reflection of how well the generated text aligns with the expected output compared to traditional metrics.

Enhanced Sensitivity: Unlike ROUGE, which primarily focuses on overlap at the token level, BERTScore provides a more granular analysis, considering the semantics of both the predictions and the ground truth. This is particularly useful in understanding the quality of generated text.

Impact on Results: The BERTScore can highlight differences in quality that might not be apparent with other metrics, giving deeper insights into the performance of the model and helping in fine-tuning for better results.

In summary, using BERTScore in evaluation provides a more comprehensive understanding of text generation quality, making it a valuable tool for assessing model performance in tasks requiring semantic accuracy.

in our example model performed well for first example and predicted well.

In [6]:
```
!jupyter nbconvert --to html New_Mid_Term_Project__Learners_notebook_(1) (2).ipynb
```

```
/bin/bash: -c: line 1: syntax error near unexpected token `('
/bin/bash: -c: line 1: `jupyter nbconvert --to html New_Mid_Term_Project__Learners_notebook_(1) (2).ipynb'
```

In [ ]:
```
```