from transformers import AutoTokenizer, AutoModelForCausalLM, pipeline

import torch

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score

# Example Dataset (for baseline comparison with traditional models)

queries = [

"How do I reset my password?",

"I want to return an item.",

"How can I cancel my order?",

"Where is my order?",

"What are your store hours?",

"Do you offer free shipping?",

"How can I track my package?",

"What is your refund policy?",

"Can I change my delivery address?",

"Do you have customer support?"

]

categories = [

"password\_reset",

"returns",

"order\_cancellation",

"order\_tracking",

"store\_info",

"shipping",

"order\_tracking",

"refund\_policy",

"delivery\_change",

"customer\_support"

]

# Split data into training and testing sets (if used for comparison models)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(queries, categories, test\_size=0.2, random\_state=42)

# Initialize the Mistral model (replace with your fine-tuned model name)

model\_name = "mistral-7b" # Replace with the name of your fine-tuned Mistral model

device = torch.device("cuda" if torch.cuda.is\_available() else "cpu") # Use GPU if available

print(f"Loading model '{model\_name}' on {device}...")

tokenizer = AutoTokenizer.from\_pretrained(model\_name)

model = AutoModelForCausalLM.from\_pretrained(model\_name).to(device)

# Create a pipeline for the Mistral model

mistral\_pipeline = pipeline("text-generation", model=model, tokenizer=tokenizer,device=0 if torch.cuda.is\_available() else -1, max\_length=150, pad\_token\_id=tokenizer.eos\_token\_id)

# Responses for categories (if fallback is needed)

responses = {

"password\_reset": "To reset your password, please visit the password reset page and follow the instructions.",

"returns": "You can return items within 30 days of purchase. Please visit our returns page for details.",

"order\_cancellation": "To cancel your order, go to your orders page and click on 'Cancel Order'.",

"order\_tracking": "To track your order, use the tracking link sent to your email or visit the 'Track Order' page.",

"store\_info": "Our store is open from 9 AM to 9 PM, Monday to Saturday.",

"shipping": "We offer free shipping on orders over $50.",

"refund\_policy": "Our refund policy allows returns and refunds within 30 days of purchase.",

"delivery\_change": "To change your delivery address, contact customer support before the order is shipped.",

"customer\_support": "You can reach customer support at support@example.com or call us at (123) 456-7890."

}

# Chatbot function using Mistral

def chatbot():

print("Welcome to the chatbot! Type 'exit' to quit.")

while True:

user\_query = input("You: ")

if user\_query.lower() == 'exit':

print("Chatbot: Goodbye!")

break

# Use the Mistral model to predict the response

mistral\_response = mistral\_pipeline(f"User: {user\_query}\nBot:")[0]["generated\_text"]

# Post-process the response to remove the prompt

response = mistral\_response.split("Bot:")[-1].strip()

# Fallback to predefined responses (if needed)

if response == "":

print(f"Chatbot: I'm sorry, I don't understand your question.")

else:

print(f"Chatbot: {response}")

# Run the chatbot

chatbot()