C]:  
  
api  
July 3, 2024  
  
from fastapi import FastAPI, HTTPException  
  
from pydantic import BaseModel  
  
from joblib import load  
  
from sklearn.metrics.pairwise import cosine\_similarity  
from fuzzywuzzy import fuzz  
  
app = FastAPI()  
  
# Load the vectorizer and classifier  
vectorizer = load('qa\_vectorizer.joblib')  
clf = load('qa\_classifier.joblib')  
  
training data = [  
  
("What is your return policy?", "Our return policy lasts 30 days."),  
  
("Tell me something about your return policy?", "Yes Sure, we would love toy,  
shelp you out.Our return policy lasts 30 days."),  
  
("How can I track my order?", "You can track your order by logging into,  
syour account."),  
  
("Do you offer international shipping?", "Yes, we offer international,  
oshipping."),  
  
("What payment methods do you accept?", "We accept credit cards (Visa,  
«MasterCard, American Express) and PayPal."),  
  
("How long does shipping take?", "Shipping usually takes 3-5 business daysy  
owithin the US."),  
  
("Can I return my order if I'm not satisfied?", "Yes, we offer a 30-day,  
oreturn policy for unused items."),  
  
("Do you provide customer support?", "Yes, our customer support team is,  
savailable 24/7."),  
("Are your products eco-friendly?", "Yes, all our products are made from,  
  
«sustainable materials."),  
  
("How do I change my account password?", "You can change your password in,  
othe account settings."),  
  
("What is your refund policy?", "We provide refunds for returned items,  
owithin 15 days of purchase."),  
  
("Do you offer discounts for bulk orders?", "Yes, we offer discounts for,  
obulk purchases. Please contact our sales team for details."),  
("Where are your products manufactured?", "Our products are manufactured,  
slocally in the United States."),  
  
("How can I contact your support team?", "You can contact our support team,  
ovia email at support@example.com or by phone at +1-123-456-7890."),  
  
("What are the shipping costs?", "Shipping costs vary based on location and,  
sorder size. You can view shipping costs at checkout."),  
  
("Is there a warranty on your products?", "Yes, we offer a 1-year warranty,  
son all our products."),  
  
("Can I cancel my order?", "You can cancel your order within 24 hours of,  
oplacing it. Please contact customer support for assistance."),  
  
("Do you have a loyalty program?", "Yes, we offer a loyalty program that,  
orewards repeat customers with discounts and special offers."),  
  
("How can I check the status of my order?", "You can check your order,  
ostatus by logging into your account and viewing your order history."),  
  
("What are your business hours?", "Our business hours are Monday to Friday,  
69:00 AM to 5:00 PM EST."),  
  
("Do you ship internationally?", "Yes, we offer international shipping to,  
omost countries. Shipping times and costs may vary."),  
  
("How do I apply a coupon code?", "You can apply a coupon code at checkout,  
sbefore completing your purchase."),  
  
("What do I do if my package is lost?", "If your package is lost, please,  
«contact our support team immediately for assistance."),  
  
("Are there any restrictions on returns?", "Items must be in unused,  
«condition with original packaging for returns. Some items may be,  
onon-returnable."),  
  
questions, answers = zip(\*training data)  
  
def calculate\_fuzzy\_match(question, trained\_question):  
return fuzz.token\_set\_ratio(question.lower(), trained\_question.lower())  
  
# Define a function to predict answers  
  
def predict\_answer (question):  
# Transform the question using the vectorizer  
question\_vector = vectorizer.transform([question] )  
  
# Calculate cosine similarity with trained questions  
similarity\_scores = cosine\_similarity(question\_vector, vectorizer.  
  
otransform(questions) ).flatten()  
  
# Calculate fuzzy match scores  
fuzzy\_scores = [calculate\_fuzzy\_match(question, q) for q in questions]  
  
# Combine both scores with equal weight  
combined\_scores = [(cosine + fuzz\_score / 100) / 2 for cosine, fuzz\_score,,  
sin zip(similarity\_scores, fuzzy\_scores)]  
  
# Get the index of the most similar question  
best\_match\_index = combined\_scores.index(max(combined\_scores) )  
best\_match\_score = max(combined\_scores)  
  
# Check if the best match score meets a certain threshold  
if best\_match\_score < 0.4: # Adjust threshold as needed  
return "Out of scope"  
  
# Return the answer of the best matching question  
return answers [best\_match\_index]  
  
class Query (BaseModel) :  
question: str  
  
@app. post ("/predict")  
async def predict(query: Query):  
question = query.question  
if not question:  
raise HTTPException(status\_code=400, detail="No question provided")  
answer = predict\_answer (question)  
return {"question": question, "answer": answer}  
if \_name.| == "\_\_main\_\_":  
import uvicorn  
uvicorn.run(app, host="0.0.0.0", port=8000)