"""

Accent-Aware Speech Recognition System

Using Deep Learning and Speaker Adaptation Techniques

for Virtual Assistant

"""

import torch

import torchaudio

import librosa

import os

import numpy as np

from transformers import Wav2Vec2ForCTC, Wav2Vec2Processor

from speechbrain.pretrained import EncoderClassifier

from flask import Flask, request, jsonify

# Initialize Flask app

app = Flask(\_\_name\_\_)

UPLOAD\_FOLDER = "uploads"

os.makedirs(UPLOAD\_FOLDER, exist\_ok=True)

# Load models

print("[INFO] Loading models...")

processor = Wav2Vec2Processor.from\_pretrained("facebook/wav2vec2-base-960h")

model = Wav2Vec2ForCTC.from\_pretrained("facebook/wav2vec2-base-960h")

model.eval()

speaker\_encoder = EncoderClassifier.from\_hparams(

source="speechbrain/spkrec-ecapa-voxceleb", savedir="pretrained\_models/spkrec"

)

# Audio preprocessing

def preprocess\_audio(file\_path, target\_sr=16000):

waveform, sr = librosa.load(file\_path, sr=target\_sr)

return waveform

# Transcription using Wav2Vec2

def transcribe(audio\_array):

input\_values = processor(audio\_array, return\_tensors="pt", sampling\_rate=16000).input\_values

with torch.no\_grad():

logits = model(input\_values).logits

predicted\_ids = torch.argmax(logits, dim=-1)

transcription = processor.decode(predicted\_ids[0])

return transcription

# Extract speaker embedding

def extract\_speaker\_embedding(file\_path):

signal, fs = torchaudio.load(file\_path)

embeddings = speaker\_encoder.encode\_batch(signal)

return embeddings.squeeze().detach().cpu().numpy()

# REST API Endpoint

@app.route("/transcribe", methods=["POST"])

def api\_transcribe():

if "audio" not in request.files:

return jsonify({"error": "No audio file provided."}), 400

file = request.files["audio"]

filename = os.path.join(UPLOAD\_FOLDER, file.filename)

file.save(filename)

try:

audio\_array = preprocess\_audio(filename)

transcription = transcribe(audio\_array)

embedding = extract\_speaker\_embedding(filename)

return jsonify({

"transcription": transcription,

"speaker\_embedding": embedding.tolist()

})

except Exception as e:

return jsonify({"error": str(e)}), 500

# Run the server

if \_\_name\_\_ == "\_\_main\_\_":

print("[INFO] Starting Flask server on port 5000...")

app.run(host="0.0.0.0", port=5000, debug=True)