# Assignment Submission

## Audio Recording, Transcription, and Text-to-Speech Pipeline

### 1.1 Overview

This project implements a complete audio processing pipeline that records audio from a microphone, transcribes the recorded audio into text using the Whisper model, and converts the transcribed text into speech using the SpeechT5 model.

### 1.2 Libraries and Models Used

- **sounddevice**: A Python library for recording and playing sound.
- **Whisper**: An automatic speech recognition (ASR) model developed by OpenAI, which is used to transcribe the recorded audio.
- **transformers**: A library from Hugging Face that provides pre-trained models for various natural language processing tasks, including text-to-speech.
- **datasets**: A library for accessing various datasets, used here to load speaker embeddings.
- **pydub**: A library for audio manipulation.

### 1.3 Parameters

- **Audio Recording**:
  - Duration: 5 seconds
  - Sample Rate: 16000 Hz
- **Whisper Model**:
  - Model Name: base.en
- **SpeechT5 Model**:
  - Model Name: microsoft/speecht5_tts
  - Vocoder: microsoft/speecht5_hifigan
- **Speaker Embeddings**:
  - Dataset: Matthijs/cmu-arctic-xvectors

Code Snippet:

```
import sounddevice as sd
import numpy as np
import whisper
from scipy.io.wavfile import write
import torch
from transformers import SpeechT5Processor, SpeechT5ForTextToSpeech, SpeechT5HifiGan
import soundfile as sf
from datasets import load_dataset
from pydub import AudioSegment

# Step 1: Record Audio
def record_audio(duration=5, sample_rate=16000, output_file="recorded_audio.wav"):
    print("Recording...")
    audio_data = sd.rec(int(duration * sample_rate), samplerate=sample_rate, channels=1, dtype='int16')
    sd.wait()
    write(output_file, sample_rate, audio_data)  # Save as WAV file
    print(f"Recording finished: {output_file}")
    return output_file

# Step 2: Transcribe Audio using Whisper
def transcribe_audio(file_path):
    model = whisper.load_model("base.en")
    result = model.transcribe(file_path)
    return result['text']

# Step 3: Convert Text to Speech using SpeechT5
def text_to_speech(text, output_file="output_speech.wav"):

    processor = SpeechT5Processor.from_pretrained("microsoft/speecht5_tts")
    model = SpeechT5ForTextToSpeech.from_pretrained("microsoft/speecht5_tts")
    vocoder = SpeechT5HifiGan.from_pretrained("microsoft/speecht5_hifigan")
    processor = SpeechT5Processor.from_pretrained("microsoft/speecht5_tts")
    model = SpeechT5ForTextToSpeech.from_pretrained("microsoft/speecht5_tts")
    vocoder = SpeechT5HifiGan.from_pretrained("microsoft/speecht5_hifigan")


    embeddings_dataset = load_dataset("Matthijs/cmu-arctic-xvectors", split="validation")
    speaker_embeddings = torch.tensor(embeddings_dataset[0]["xvector"]).unsqueeze(0)


    inputs = processor(text=text, return_tensors="pt")


    speech = model.generate_speech(
        inputs["input_ids"],
        speaker_embeddings=speaker_embeddings,
        vocoder=vocoder
    )


    sf.write(output_file, speech.numpy(), 16000)
    print(f"Speech saved as: {output_file}")


def main():
    # Step 1: Record Audio
    audio_file = record_audio(duration=5)  # Record 5 seconds of audio

    # Step 2: Transcribe Audio
    transcription = transcribe_audio(audio_file)
    print("Transcription:", transcription)

def main():
    # Step 1: Record Audio
    audio_file = record_audio(duration=5)  # Record 5 seconds of audio

    # Step 2: Transcribe Audio
    transcription = transcribe_audio(audio_file)
    print("Transcription:", transcription)

    # Step 3: Convert Transcribed Text to Speech
    text_to_speech(transcription)

if __name__ == "__main__":
    main()
```

### Step 1: Record Audio

This step captures audio input from the microphone and saves it as a WAV file. We use the sounddevice library to handle audio recording.

1. **Import Libraries**: Import necessary libraries for recording (sounddevice) and saving audio files (scipy.io.wavfile).
2. **Function Definition**: Define the record_audio function that takes duration, sample_rate, and output_file as parameters.
3. **Recording**: Use sd.rec() to record audio data for the specified duration at the given sample rate.
4. **Wait for Completion**: sd.wait() pauses execution until the recording is complete.
5. **Save Audio**: Use write() to save the recorded audio to a WAV file.
6. **Return File Path**: The function returns the path of the saved audio file.

**Step 2: Transcribe Audio**

1. **Import Library**: Import the Whisper library.
2. **Function Definition**: Define the transcribe_audio function that accepts the file_path of the recorded audio.
3. **Load Model**: Load the Whisper model using whisper.load_model().
4. **Transcribe**: Use model.transcribe() to transcribe the audio file into text.
5. **Return Transcription**: The function returns the transcribed text from the audio.

**Step 3: Convert Text to Speech**

This step converts the transcribed text into speech using the SpeechT5 model. The SpeechT5 model requires speaker embeddings to generate more realistic speech.

1. **Import Libraries**: Import necessary libraries for text-to-speech processing and loading datasets.
2. **Function Definition**: Define the text_to_speech function, which takes text and output_file as parameters.
3. **Load Processor and Models**: Load the SpeechT5 processor, model, and vocoder using from_pretrained().
4. **Load Speaker Embeddings**: Use load_dataset() to load speaker embeddings from a specified dataset.
5. **Prepare Input**: Use the processor to convert the text into input tensors suitable for the model.
6. **Generate Speech**: Call generate_speech() to generate speech from the input text using the model, vocoder, and speaker embeddings.
7. **Save Output**: Use sf.write() to save the generated speech as a WAV file.

**Step 4 :Main Function to Run the Complete Pipeline**

1. **Main Function Definition**: Define the main function to encapsulate the entire workflow.
2. **Record Audio**: Call record_audio() to record audio for 5 seconds and store the returned file path.

3. **Transcribe Audio**: Call transcribe_audio() to transcribe the recorded audio and print the transcription.
4. **Convert to Speech**: Call text_to_speech() to convert the transcribed text into speech.
5. **Execution**: The if __name__ == "__main__": block ensures that the main function is executed when the script runs.

**Link of recording to see implementation:**

**https://drive.google.com/file/d/1Ib9MNod5p2v_RmIgmXBa56p-jpiUN63m/view?usp=drive_link**

Submitted By : Shreya Dhanrao

Mno : 7745828578

Email: shreyadhanrao23@gmail.com