AI Voice Assistance Pipeline

Christo John

August 27, 2024

Contents

1 Introduction

This project focuses on processing an audio file to transcribe its content, filter out non-speech parts, process the transcribed text, and convert the processed text back into speech. The project is organized into four primary modules: Audio Transcription, Voice Activity Detection (VAD), Text Processing, and Text-to-Speech Conversion.

2 Models Used

2.1 Whisper

Whisper is an automatic speech recognition (ASR) model used for transcribing spoken language into text. It is known for its ability to handle noisy audio and various accents.

2.2 LLaMA

LLaMA (Large Language Model) is a model designed for natural language understanding and generation. It excels in tasks such as text generation, summarization, and answering questions, making it suitable for various NLP applications.

2.3 Edge TTS

Edge TTS is a text-to-speech synthesis model that converts written text into spoken words. It provides high-quality, natural-sounding speech.

3 Dependencies

- **whisper** - For audio transcription.
- **llama** - For audio processing.
- **edge-tts** - For text-to-speech conversion.
- **groq-api** - For advanced natural language processing tasks.
- **FFmpeg** - For audio processing.

4 System Architecture

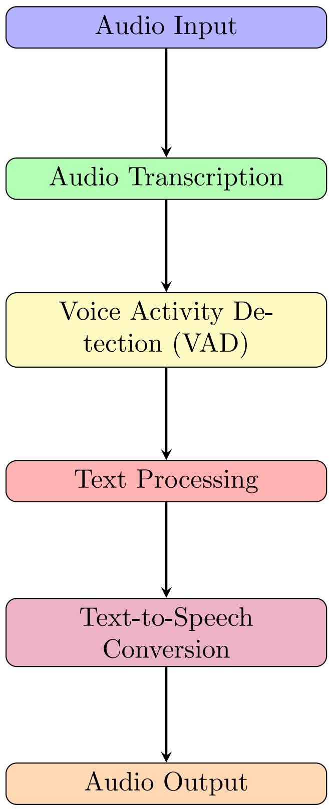

Figure 1: System Architecture Overview

5 Modules Overview

5.1 Module 1: Audio Transcription

Transcribe audio files into text. It begins by loading the Whisper model and taking in an audio file specified by its path. The model then processes the audio and outputs the corresponding transcribed text, serving as the basis for further processing.

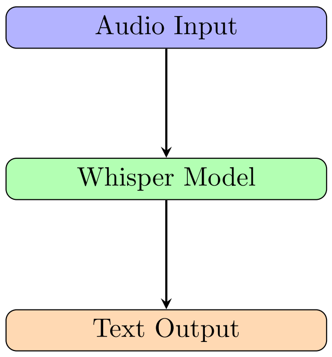

Figure 2: Audio Transcription Process

5.2 Module 2: Voice Activity Detection (VAD)

System processes the audio file to identify and isolate segments of voice activity. The audio is first segmented into chunks, with each chunk's RMS value being calculated to determine whether it contains speech. The module filters out non-speech parts and, if necessary, resamples the audio to 16kHz. The output is an audio segment containing only the relevant voice activity, ensuring that only important parts of the audio are further processed.

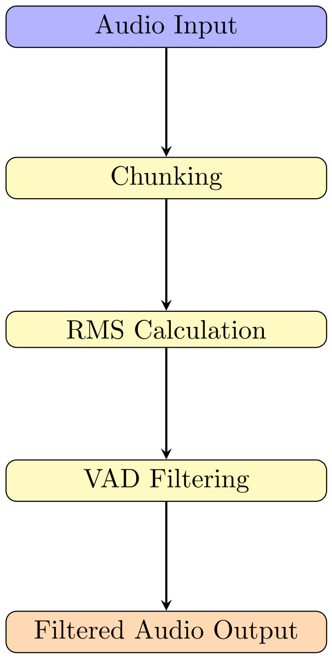

Figure 3: Voice Activity Detection Process

5.3 Module 3: Text Processing using Groq API and Llama Model

Handles advanced text processing tasks. After receiving the transcribed text, it sends the text to the Groq API, which utilizes the Llama model to perform operations such as summarization, translation, or sentiment analysis. The processed text returned by the Groq API is refined and ready for the next stage of the pipeline.

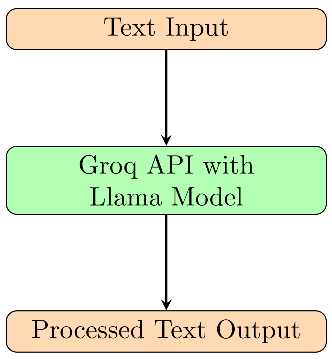

Figure 4: Text Processing Using Groq API with Llama Model

5.4 Module 4: Text-to-Speech Conversion

Converts the processed text into speech. Using the Edge TTS model, it synthesizes the text and generates an audio file. The resulting speech is saved to a specified output path, completing the transformation from raw audio input to refined audio output.

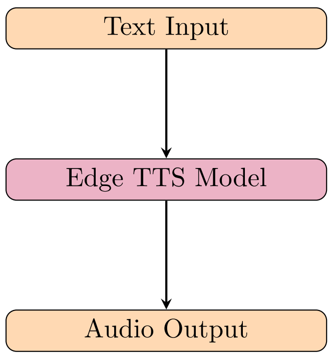

Figure 5: Text-to-Speech Conversion Process

6 Code

```
import whisper
from pydub import AudioSegment
from groq import Groq
from google.colab import userdata
import edge_tts
import asyncio

# Load Whisper model
model = whisper.load_model("base")

# Load the audio file
audio = AudioSegment.from_file("/content/Recording3.wav")

# Function to calculate RMS (Root Mean Square) of an audio segment
def rms(segment):
    return segment.dBFS / 10 ** (segment.frame_rate / 20 * 0.05)

# Resample the audio to 16kHz if necessary
if audio.frame_rate != 16000:
    audio = audio.set_frame_rate(16000)
    print("Resampled to 16kHz")

# Define the chunk duration in milliseconds
chunk_duration_ms = 500

# Calculate the number of samples per chunk
chunk_samples = int(chunk_duration_ms / 1000 * 16000)

# Calculate the RMS values for each chunk
rms_values = [rms(audio[i:i + chunk_samples]) for i in range(0, len(audio), chunk_samples)]

# Set the voice activity detection (VAD) threshold
vad_threshold = 0.5  # Adjust the threshold as needed

# Determine voice activity based on RMS values and threshold
voice_activity = [rms_value > vad_threshold for rms_value in rms_values]

# Filter out non-voice activity chunks
chunks = [audio[i:i + chunk_samples] for i, active in enumerate(voice_activity) if active]

# Combine the filtered chunks into one audio segment
filtered_audio = sum(chunks) if chunks else audio

# Save the filtered audio (optional)
```

```
filtered_audio.export("/content/input/filtered_audio.wav", format="wav")
# print("Filtered audio saved as filtered_audio.wav")

# Transcribe the filtered audio
result = model.transcribe("/content/input/filtered_audio.wav")
transcribed_text = result["text"]

# Process the transcribed text using Groq API
groq_api_key = userdata.get('GROQ_API_KEY')
client = Groq(api_key=groq_api_key)

chat_completion = client.chat.completions.create(
    messages=[{"role": "user", "content": transcribed_text}],
    model="llama3-8b-8192",
    n=2,
)
processed_text = chat_completion.choices[0].message.content

# Convert the processed text to speech using edge_tts
async def text_to_speech(text, output_file):
    communicate = edge_tts.Communicate(text, 'en-US-JennyNeural', pitch='+200Hz')
    try:
        await communicate.save(output_file)
        print(f"Saved {output_file}")
    except edge_tts.exceptions.NoAudioReceived as e:
        print(f"Error: {e}")
        print(f"Text sent: {text}")

# Run the text-to-speech conversion
output_file = "/content/output/test.mp3"
await text_to_speech(processed_text, output_file)
```

7 Conclusion

A robust pipeline for audio processing, from real-time voice activity detection to speech synthesis. It uses the Pydub library for filtering audio segments based on RMS values, followed by transcription with the Whisper model. The transcribed text is then processed using the Groq API with the Llama model for advanced text tasks. While the current implementation handles audio and text through file storage, future enhancements may include streaming these components to streamline the process further.

8 Future Scope

- **Enhanced VAD** - Implement more advanced voice activity detection techniques for improved accuracy.
- **Support for Additional Languages** - Extend support for more languages and accents in both transcription and text-to-speech conversion.
- **Real-time Processing** - Develop capabilities for real-time audio processing and speech synthesis.
- **Integration with Other Services** - Integrate with additional APIs for enhanced text processing features like sentiment analysis and summarization.
- **User Interface** - Create a user-friendly interface for easier interaction with the audio processing system.