Simon Tesfatsion

GTTS Speech to Text to Speech

GItHUB LInk

Google Slides Link

Introduction

In this project, I developed an interactive assistant capable of:

- Converting user speech to text.
- Processing the text using OpenAI's GPT model for intelligent responses.
- Converting the generated text into speech for conversational interaction.

Environment Setup

Backend:

- Python-based backend using Flask for API endpoints.
- Key Python libraries:
  - Speech Recognition and OpenAI's Whisper for speech-to-text.
  - gTTS for text-to-speech.

Installation:

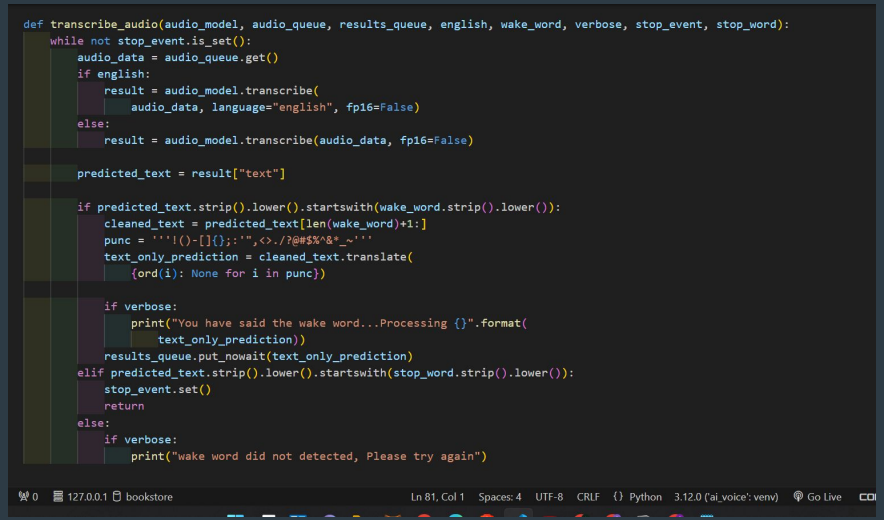
  - Backend: pip install -r requirements.txt

Backend

- Key Responsibilities:
  - Handle audio files sent from the frontend.
  - Convert audio to text using the Whisper model.
  - Generate responses using OpenAI's GPT model.
- Tools and Libraries:
  - Flask for API endpoints.
  - Queue for handling audio processing tasks.
  - Multithreading for real-time processing.

Speech to Text

- Library: OpenAI’s Whisper model.
- Process:
  - Accepts audio from the frontend.
  - Converts it into text using Whisper's high-accuracy transcription.
- Code :



```
def transcribe_audio(audio_model, audio_queue, results_queue, english, wake_word, verbose, stop_event, stop_word):
    while not stop_event.is_set():
        audio_data = audio_queue.get()
        if english:
            result = audio_model.transcribe(
                audio_data, language="english", fp16=False)
        else:
            result = audio_model.transcribe(audio_data, fp16=False)

        predicted_text = result["text"]

        if predicted_text.strip().lower().startswith(wake_word.strip().lower()):
            cleaned_text = predicted_text[len(wake_word)+1:]
            punc = '''!()-[]{};:'",<>./?@#$%^&*_~'''
            text_only_prediction = cleaned_text.translate(
                {ord(i): None for i in punc})

            if verbose:
                print("You have said the wake word...Processing {}".format(
                    text_only_prediction))
            results_queue.put_nowait(text_only_prediction)
        elif predicted_text.strip().lower().startswith(stop_word.strip().lower()):
            stop_event.set()
            return
        else:
            if verbose:
                print("wake word did not detected, Please try again")
```

LLM Response

- Library: OpenAI GPT model via OpenAI API.
- Process:
  - The transcribed text from Whisper is sent as a query to the GPT model.
  - The model generates a conversational and contextually appropriate response.
- Code :

```
def reply(llm, stop_event, results_queue):
    while not stop_event.is_set():
        result = results_queue.get()
        reponse = llm.chat.completions.create(
            model="gpt-4o", messages=[
                {"role": "system",
                 "content": """You are helpfull voice assistant, Your task is to
                 understand what the transcribed text is talking about and give a valid response.
                 if You didn't understand what the user is asking politly ask them to clarify thier question.
                 give the output in plain english"""},
                {"role": "user", "content": result}], temperature=0, max_tokens=100)
        answer = reponse.choices[0].message.content
```

Text to Speech

- Library: Google Text-to-Speech (gTTS)
- Process:
  - The response text from GPT is converted into speech using gTTS.
  - The audio is sent back to the frontend for playback.
- Code :

GTTS

```
mp3_obj = gTTS(text=answer, lang="en", slow=False)
mp3_obj.save("answer.mp3")
reply_audio = AudioSegment.from_mp3("answer.mp3")
play(reply_audio)
```

Project Reference Materials

GitHub Link : https://github.com/Montegan/SFBU_STT_TTS

Google Slides Link :

https://docs.google.com/presentation/d/1PcBKRv02_5PSOLjlq8s1aB3mt4Sbvcec_yJDcOPLMeI/edit?usp=sharing

Thank You

Simon Tesfatsion