Simon Tesfatsion

Openai TTS
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.
  - oTTSfor text-to-speech.

Installation:

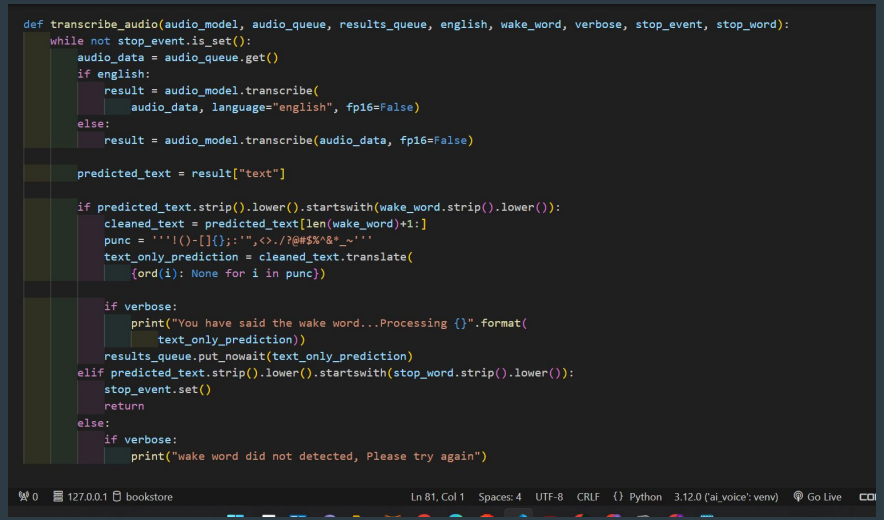
  - Backend: pip install -r requirements.txt

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: Openai TTS.
- Process:
  - The response text from GPT is converted into speech using gTTS.
  - The audio is sent back to the frontend for playback.
- Code :

OTTS

```
mp3_obj = llm.audio.speech.create(
    model="tts-1", voice="alloy", input=answer)  # type: ignore
mp3_obj.stream_to_file("reply.mp3")
reply_audio = AudioSegment.from_mp3("reply.mp3")
play(reply_audio)
os.remove("reply.mp3")
```

Demo

```
(ai_voice) PS C:\Users\H00422003\Desktop\SFBU\2ndsem\GenAI\week_10_homework\backend> python openai_assistant.py  --model base --english --energy 300  --pause 0.8 --dynamic_energy --wake_word "hey computer"     --verbose True
C:\Users\H00422003\Desktop\SFBU\2ndsem\GenAI\week_10_homework\backend\ai_voice\Lib\site-packages\whisper\__init__.py:150: FutureWarning: You are using `torch.load` with `weights_only=False` (the current default value), which uses the default pickle module implicitly. It is possible to construct malicious pickle data which will execute arbitrary code during unpickling (See https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models for more details). In a future release, the default value for `weights_only` will be flipped to `True`. This limits the functions that could be executed during unpickling. Arbitrary objects will no longer be allowed to be loaded via this mode unless they are explicitly allowlisted by the user via `torch.serialization.add_safe_globals`. We recommend you start setting `weights_only=True` for any use case where you don't have full control of the loaded file. Please open an issue on GitHub for any issues related to this experimental feature.
  checkpoint = torch.load(fp, map_location=device)
Listening...
tensor([0., 0., 0.,  ..., 0., 0., 0.])
tensor([ 0.0000e+00,  0.0000e+00, -3.0518e-05,  ..., -3.0518e-04,
        -3.3569e-04, -9.1553e-05])
tensor([ 0.0000e+00,  0.0000e+00, -3.0518e-05,  ...,  2.7466e-04,
         3.0518e-04,  2.7466e-04])
You did not say the wake word.. Ignoring
tensor([-2.1362e-03, -1.9226e-03, -1.4038e-03,  ...,  0.0000e+00,
```

Project Reference Materials

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

Google Slides Link :

https://docs.google.com/presentation/d/1o2PgFdpUe3v2ttnkG-mrkHmmzWrvPlteKHJ1yHOfCGM/edit?usp=sharing

Thank You

Simon Tesfatsion