Real-time Speech to Text to Speech : Building Your AI-Based Alexa

Using OpenAI-Whisper and gTTS

Table of content

Introduction

- The project is created which functions as a voice assistant powered by OpenAI's GPT-3.5 Turbo model, combining various components to create an AI-based voice assistant.
- The process involves waiting for a wake word “hey computer”, converting the user's spoken input into text using OpenAI Whisper(speech to text), and generating responses through the chatGPT API.
- The aim is to enhance voice assistants beyond the limitations of existing systems like Alexa.
- The goal is to enable the assistant to handle complex questions and offer valuable information, envisioning a more intelligent and capable voice assistant experience.

Design

1. The user asks a question using their microphone.
    a. We will use Python SpeechRecognition[46] to record the Question.
2. OpenAI Whisper automatically transcribes the question into text.
3. The text is passed to OpenAI GPT completions endpoints.
4. The OpenAI API returns an answer.
5. The answer is saved to an mp3 file and passed to Google Text to Speech (gTTS) to create a voice response.

Implementation steps in details

Step1: Recording the Audio

- **Input:**
  - Parameters: audio_queue, energy, pause, dynamic_energy.
- **Process:**
  - Utilizes SpeechRecognition to create a recognizer.
  - Infinite loop captures and converts audio to PyTorch tensor.
  - Normalizes and stores tensor in audio_queue.
- **Output:**
  - Recorded audio in PyTorch tensor format is placed in audio_queue.

Implementation steps in details cont..

Step2: Transcribe the Audio

- **Input:**
  - Parameters: audio_queue, result_queue, audio_model, english, wake_word, verbose.
- **Process:**
  - Retrieves audio data from audio_queue.
  - Transcribes audio using audio_model with language detection or English transcription based on the english flag.
  - Extracts and processes the transcribed text, removing the specified wake_word and punctuation.
  - If verbose, prints a message about wake word detection and text processing.
  - Adds the processed text to result_queue.
- **Output:**
  - Processed text is stored in result_queue.

Implementation steps in details cont..

Step3: Replying to user Request

- **Input:**
  - Parameter: result_queue.
- **Process:**
  - Retrieves result from result_queue.
  - Language Model Call:Uses OpenAI's language model (Davinci) to generate a response.
  - Text-to-Speech Conversion:
    - Converts the generated response to an audio file using gTTS.
    - Saves the audio as "reply.mp3."
  - Audio Playback:Plays back the "reply.mp3" file.
  - File Cleanup:Deletes "reply.mp3."
- **Output:**
  - Generates and plays back an audio response from ChatGPT.

Implementation

- Install the following supported packages

  pip install pydub pyaudio speechrecognition whisper torch numpy gtts openai click

- Run the Script
  Python3 my_app.py
- Ask Question. It continuously listening and awake the system using “hey Computer ” wake word
- If you want to stop running say “stop”
- It gives output in response.txt and .mp3 (speech) file

Test

response1.txt

```
User's question:  how are you
AI's response: Hello there! I'm doing well, thank you for asking. How are you?
```

response.txt

```
User's question:  what is the weather outside
AI's response: The weather outside depends on your location. You can check the current
weather conditions in your area by visiting a weather website. Generally, the weather
outside is either hot, cold, sunny, rainy, or a combination of these. You can also check
the forecast for the week to get an idea of what the weather might be like in the upcoming
days. Additionally, you can use a weather app to get real-time updates about the current
temperature, wind speed, humidity, and other weather conditions.
```

Enhancement

- Give URL , Text or video as input generates vector based on this and then Question ans based on that using speech.
- Implement to manage Robot through commands and work according to the commands
- Streaming audio replies in real-time instead of storing them on disk.
- Incorporating a caching mechanism for optimization.
- Introducing a stop word for prompt interruption without delay.

Conclusion

- In summary, the project introduces a voice assistant powered by OpenAI's GPT-3.5 Turbo model, aiming to surpass the limitations of existing systems like Alexa. The process involves recognizing a wake word, transcribing user input with OpenAI Whisper, and generating responses through the ChatGPT API. The goal is to create an intelligent voice assistant capable of handling complex questions. The system records user questions, transcribes them using Python SpeechRecognition, and employs OpenAI models to provide answers, creating a seamless voice interaction experience.

Github link

https://github.com/DKruti/Machine-Learning/tree/master/Generative%20AI/Speech-to-text-to-speech%20AI%20supported

References

- https://hc.labnet.sfbu.edu/~henry/sfbu/course/generative_ai/Building_Your_AI_Based_Alexa/slide/exercise_Building_Your_AI_Based_Alexa.html
- https://hc.labnet.sfbu.edu/~henry/sfbu/course/generative_ai/Building_Your_AI_Based_Alexa/slide/Introduction.html
- https://hc.labnet.sfbu.edu/~henry/sfbu/course/generative_ai/Building_Your_AI_Based_Alexa/slide/The_Main_Function.html#run