Voice-Activated AI Assistant Using OpenAI and Whisper

A Real-Time Voice Interaction System

Ranishree Anegundi

Overview

Objective:

- - Create a voice-activated assistant capable of:
- - Speech recognition
- - Natural language processing
- - Voice-based response generation

Key Features:

- - Real-time audio transcription
- - Wake-word detection
- - Integration with OpenAI's ChatGPT
- - Text-to-speech response system

System Architecture

Components:

- 1. Audio Recording: Captures user audio input.
- 2. Speech Recognition: Transcribes audio using Whisper model.
- 3. Wake-Word Detection: Detects specific trigger words to activate the assistant.
- 4. NLP Processing: Sends the transcribed query to ChatGPT.
- 5. Text-to-Speech (TTS): Converts the AI response to speech.

Flow Diagram:

- User speaks → Audio Recording → Whisper Transcription → Wake Word Detection → ChatGPT Query → TTS Response → User hears response.

Tools and Technologies

Programming Language: Python

- - Libraries and Frameworks:
- - torch, numpy: Data handling and processing.
- - speech_recognition: Audio input handling.
- - whisper: OpenAI's speech-to-text model.
- - gTTS and pydub: Text-to-speech and audio playback.
- - openai: Interact with ChatGPT.

Hardware Requirements:

- - Microphone
- - GPU (optional for Whisper model optimization)
- - Environment:
- - .env file to securely load OpenAI API keys.

Code Walkthrough - Main Modules

Audio Recording:

- - Captures audio continuously.
- - Uses speech_recognition for microphone input.

Transcription and Wake Word Detection:

- - Transcribes speech using Whisper.
- - Detects wake word before processing queries.

NLP with ChatGPT:

- - Sends the query to OpenAI's GPT-3.5-Turbo model.
- - Receives a natural language response.

Text-to-Speech Response:

- - Uses gTTS for speech synthesis.
- - Plays the response audio using pydub.

Code Highlights

Wake Word Detection:

```
if predicted_text.lower().startswith(wake_word.lower()):
    # Strip wake word and send query
```

ChatGPT Integration:

```
payload = {
    'model': 'gpt-3.5-turbo',
    'messages': [{'role': 'user', 'content': prompt}],
    'max_tokens': 300,
    'temperature': 0.7,
}
response = requests.post('https://api.openai.com/v1/chat/completions', headers=headers, json=payload)
```

- Speech Synthesis:

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

Demo Workflow

Step 1: Start the program.

Step 2: Say the wake word (e.g., 'Hey Computer').

Step 3: Ask a question or give a command.

Step 4: Wait for the AI to respond with synthesized speech.

Challenges and Solutions

Challenges:

- - Real-time audio processing.
- - API communication delays.
- - TTS performance and accuracy.

Solutions:

- - Efficient threading for parallel tasks.
- - Optimized Whisper model for quick transcription.
- - Use of gTTS and pydub for seamless audio playback.

Future Enhancements

Improvements:

- - Multilingual support.
- - Advanced wake-word models using machine learning.
- - Enhanced TTS quality with neural-based models.
- - Integration with IoT devices for smart home automation.

Conclusion

Summary:

- - Successfully implemented a voice-activated assistant.
- - Combines cutting-edge speech recognition and NLP.
- - Practical application in smart assistants and automation.

Questions?