Speech-to-speech System

-Bob

Shreya Sari
9440830434
shreyasari24@gmail.com

Abstract

This Speech-to-Speech System utilizes speech recognition and text-to-speech technologies to convert spoken language into text, process it, and synthesize responses. It enables natural interactions through predefined queries, context maintenance, and continuous learning, making it ideal for personal assistants and customer service applications while enhancing user communication experiences.

Libraries used

- speech_recognition
- pyttsx3
- langchain_ollama
- langchain_core

Functions of libraries used

- **speech_recognition:** Main Function: Provides functionalities to recognize speech from audio input, allowing the program to convert spoken language into text. It supports various speech recognition engines and enables real-time audio processing.

- **pyttsx3:** Main Function: Converts text into speech (text-to-speech). It allows the application to vocalize responses, making it interactive by reading out the bot's replies using the built-in speech engines of the operating system.

- **langchain_ollama:** Main Function: Facilitates interaction with large language models (LLMs) like LLaMA. It enables the construction of conversational AI by managing prompts and context effectively, allowing the model to generate coherent responses based on the conversation history.

- **langchain_core:** Main Function: Provides tools for creating and managing prompts for LLMs. It helps in structuring the conversation flow and enables the chaining of different components together, enhancing the modularity and reusability of the code.

Features

- Speech Recognition
- Predefined Responses
- Dynamic Conversation Handling
- Text-to-Speech Conversion
- Continuous Interaction
- Noise Adjustment
- Wake word implementation

Flow of the project

- **Speech Recognition:** Start a listener to detect audio input.
- **Wake Word Detection:** Continuously listen for a predefined wake word (e.g., "Hey Bob") to activate the conversation.
- **Response Generation:** Check if the user's question matches any predefined questions for quick responses.
- **Text-to-Speech Conversion:** Convert the generated response (text) into speech using a text-to-speech library (e.g., pyttsx3).
- **Continuous Interaction:** Allow for ongoing dialogue by returning to the input handling step after each response. Provide an option to exit the conversation by listening for a specific command (e.g., "exit").
- **Termination:** Gracefully terminate the program upon receiving the exit command.

Challenges and Solutions

- **Background Noise Interference**
  Solution: Use noise-canceling microphones and implement noise filtering techniques.

- **Speech Recognition Errors**
  Solution: Use a more accurate speech recognition service and provide clear user instructions.

- **Latency in Response**
  Solution: Optimize the code and use asynchronous processing for faster responses.

- **Limited Vocabulary**
  Solution: Continuously update predefined questions and leverage machine learning to expand vocabulary.

- **System Compatibility Issues**
  Solution: Test the system on multiple platforms and provide user guidance for audio setup.

- **Resource Limitations**
  Solution: Optimize models for efficiency or leverage cloud services for processing.

Thank you