Virtual Voice Assistant - ASTRA

Members: Ritik Soni (241149)
Krish Shrimali (241145)
Adithya Rai (241136)
Shivam Makwana (241115)
Course: B.Tech (AI-ML)
Div : B
College: JG University

1. Introduction

The 'Virtual Voice Assistant - Astra' is a Python-based voice-controlled application that performs basic desktop automation tasks using natural voice commands. It utilises speech recognition and text-to-speech technologies to interact with the user and execute commands, such as opening applications, performing web searches, displaying the current time, and putting the system to sleep.

2. Objectives

The main objectives of this project are:

- To create a voice-based interface that interacts naturally with the user.
- To automate common desktop operations using voice commands.
- To apply Python libraries for speech recognition and text-to-speech conversion.
- To develop a foundation for future AI-based personal assistants.

3. Tools and Technologies Used

- Programming Language: Python
- Libraries: speech_recognition, pyttsx3, datetime, os, webbrowser, time
- Platform: Windows
- IDE: Visual Studio Code

4. Working of the Project

The AI Voice Assistant continuously listens to the user's voice commands using a microphone. It converts the speech into text using the SpeechRecognition library, processes the command, and performs an appropriate action such as opening a website, telling the current time, or putting the computer to sleep. The pyttsx3 library is used to convert the assistant's responses from text to speech, creating an interactive experience.

5. Flow of Execution

1. The system initializes the text-to-speech engine.
2. The assistant greets the user and starts listening.
3. When a command is detected, it is converted to text.
4. The command is analyzed to determine which action to perform.
5. The assistant executes the corresponding function and responds verbally.
6. The process continues until the user says 'exit' or 'quit'.

6. Key Features

- Voice-controlled interaction.
- Opens websites and desktop applications.
- Provides current date and time.
- Can put the computer into sleep mode.
- User-friendly and extendable codebase.

7. Code Overview

The code consists of several key functions:

- speak(text): Converts text to speech using pyttsx3.
- take_command(): Listens through the microphone and converts speech to text.
- execute_command(query): Executes the action based on recognized command.

The main loop continuously listens for user input until an exit command is received.

8. Output and Results

The assistant accurately recognises commands and performs the corresponding actions. It provides audio responses for user feedback, ensuring smooth interaction. For example:

- 'Open YouTube' → Launches YouTube in a web browser.
- 'What time is it' → Responds with the current system time.
- 'Go to sleep' → Puts the computer into sleep mode.
- 'Exit' → Closes the assistant gracefully.

9. Future Scope

This project can be extended by integrating Artificial Intelligence and Natural Language Processing (NLP) to improve command understanding. Additional features like email automation, reminders, and GUI integration can make the assistant more powerful and user-friendly.

10. Conclusion

The 'AI-Powered Desktop Voice Assistant' demonstrates the power of Python in developing intelligent and interactive desktop applications. It provides a foundation for more advanced AI assistants and helps in understanding the working of voice-based systems.