International Journal of Advanced Multidisciplinary Research and Educational Development
Volume 2, Issue 1 | January - February 2026 | www.ijamred.com
ISSN: 3107-6513

Lightweight Rule Based Voice Assistant

Vaishnavi Gujar, Vrushali Randhave, Sanika Helkar, Asmita Nage, Atharva Siddhanti

Department of Computer Engineering, JSPM'S Rajarshi Shahu College of Polytechnic, Pune, Maharashtra, India

gujarvaishnavi35@gmail.com

Abstract:

The **Lightweight rule-based voice assistant** project is designed to create a lightweight, command-line-driven virtual assistant that operates without machine learning models. The core objective is to build a transparent and predictable system for handling user queries through a predefined set of rules.

The system operates by taking text-based input from the user through a simple command-line interface. It leverages Python's native data structures and the re module for regular expression matching to power its custom rule engine. This engine parses user input to identify keywords and patterns that match its knowledge base.

The knowledge base is a simple, human-readable structure (such as a dictionary or JSON file) that maps specific input patterns to corresponding outputs or actions. This design allows for easy expansion and customization without the need for model retraining.

The assistant's capabilities are mapped to handle a variety of specific tasks, including answering frequently asked questions, performing basic calculations, retrieving predefined information, and executing simple commands. The application provides a fast, efficient, and highly reliable method for automating responses and tasks in a controlled environment.

Keywords: Rule-Based System, Offline Voice Assistant, Python, Speech Recognition, Text-to-Speech

1.Introduction:

In the world of conversational AI, most systems rely on complex machine learning models that often act like "opaque systems" and require heavy computational resources. While powerful, they can be unpredictable and hard to understand. To address this, we designed the **Rule-Grounded Python Voice Assistant**, a lightweight and deterministic system that handles tasks through a clear set of rules.

Using Python's **Speech Recognition** library for voice input and **pyttsx3** for spoken output, the assistant converts voice commands into precise actions. It processes the transcribed text with a custom rule engine, identifying keywords and patterns to match a simple, human-readable knowledge base (like a dictionary or JSON file). This allows easy expansion without retraining.

The assistant can answer frequently asked questions, perform basic calculations, and retrieve predefined information, providing a fast, reliable, and voice-driven interface voice interface for automating routine tasks.

2.LITERATURE SURVEY

Voice assistants have become a common part of everyday computing, allowing users to interact with systems through natural speech and perform tasks without using a keyboard or mouse. Popular assistants such as **Google Assistant, Amazon Alexa, Apple Siri, and Microsoft Cortana** use advanced artificial intelligence and cloud-based architectures to deliver a wide range of features. While these systems are powerful, they rely heavily on internet connectivity, which limits their usability in offline environments and raises concerns related to user privacy and data security. In addition, the high computational demands of AI-based processing can reduce performance on devices with limited resources.

Examples:

- **Google Assistant:** Provides extensive functionality but requires continuous internet access, as user commands are processed on cloud servers.
- **Amazon Alexa:** Widely used for smart home automation and third-party services, yet it depends almost entirely on cloud processing.
- **Apple Siri:** Combines on-device and cloud-based processing, offering limited offline support, though most advanced features still require connectivity.
- **Microsoft Cortana:** Integrated into Windows, supports basic tasks such as reminders and searches but offers minimal offline capabilities.

These challenges have encouraged research into alternative solutions. **Offline speech recognition libraries** such as Voski and Pocket Sphinx allow voice commands to be processed locally, improving response time and privacy. Open-source platforms like Mycroft and Jasper demonstrate offline voice assistants but often involve complex setup procedures. Rule-based systems, which map predefined commands to actions, offer fast and predictable execution, making them suitable for controlled environments, though they lack flexibility. Hybrid architectures that combine rule-based logic with lightweight

ISSN: 3107-6513

adaptive components provide a balanced approach, offering transparency, reliability, and offline functionality.
Overall, the literature shows that although AI-driven voice assistants are versatile, their cloud dependence limits privacy and offline use. This highlights a research gap for a **PC-based hybrid voice assistant** that integrates offline speech recognition with rule-based command processing to deliver a secure, efficient, and user-friendly voice-controlled system

3.METHODOLOGY:

The proposed methodology outlines the design and implementation of a rule-based offline voice assistant for personal computers. The system emphasizes offline speech recognition, deterministic rule-based command execution, and text-to-speech feedback to deliver a fast, interactive, and privacy-focused user experience.
The methodology follows a structured sequence of stages:

1. Voice input is captured through a microphone connected to the PC. The audio is recorded in real time and forwarded for processing.
2. The captured audio is converted into text using offline speech recognition libraries such as Voski or Pocket Sphinx, ensuring that all processing occurs locally without internet dependency.
3. The recognized text is then analyzed using a rule-based command processor. The system compares the input text with predefined commands stored in a dictionary or configuration file, where each rule is mapped to a specific system action. For example, the command “Open Notepad” triggers the execution of **notepad.exe**.
4. Once a valid command is identified, the corresponding action is executed using Python modules such as so, subprocess, or **PyAutoGUI**.
5. After execution, the system provides audible feedback to the user using an offline text-to-speech engine like **pyttsx3**, confirming successful task completion.

Overall Benefits:

- 100% Offline & Private: Processes commands locally for instant responses and complete data security.
- Simple & Customizable: Built on a straightforward, rule-based engine that allows easy addition or modification of commands.
- Lightweight & Efficient: Designed to run smoothly on low-power hardware.

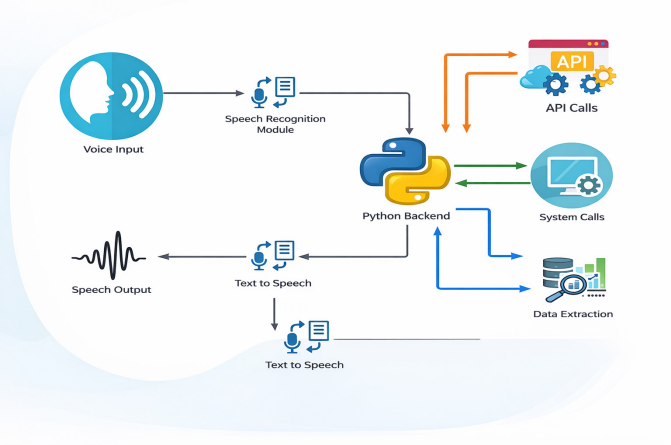

Fig. 1: Workflow of Proposed System

4. HARDWARE LIST

1.Personal Computer / Laptop - Development and testing environment.
2.Microphone - To capture user voice commands.
3.Speakers / Headphones -To provide audio feedback via text-to-speech 4. Internet (Optional for initial setup)-To download required libraries and dependencies
5. Storage (HDD/SSD)-To store project files, audio data, and software.

5. SOFTWARE LIST

1.Python (3.x) -Core programming language
2.IDE (PyCharm / Visual Studio Code) -Writing and debugging code
3.Vosk / PocketSphinx - Offline speech recognition
4.pyttsx3 - Offline text-to-speech conversion
5.os / subprocess / PyAutoGUI - Executing system-level commands
6.Operating System: Windows/Linux Platform - for development and execution.

6.Advantages and Disadvantages

Advantages:

- Interacting with the system through voice feels natural and user-friendly.
- Tasks can be completed faster since commands are given directly by voice.

 ISSN: 3107-6513

- Users can operate the system without touching keyboards or mice.
- The predefined rules make command recognition consistent and dependable.

Disadvantages:

- Background noise or poor microphone quality can reduce accuracy.
- Converting speech to text may slow down performance on some computers.
- The system may not understand different accents or alternative phrasings well.
- Adding new voice commands requires manual changes to configuration files.
- Handling errors like microphone issues or failed transcription could be improved.

7. Implementation:

The system is designed as a simple, fast, and privacy-focused offline voice assistant for personal computers.

- It captures user voice commands through a microphone connected to the PC and processes them in real time.
- Voice inputs are converted into text using offline speech recognition tools like **Vosk** or **Pocket Sphinx**, so no internet connection is required.
- The converted text is analyzed by a custom Python-based rule engine, which matches the commands against a set of predefined rules stored in a dictionary or JSON file.
- Each rule corresponds to a specific action, such as opening applications, performing basic calculations, or retrieving stored information.
- Regular expressions are used to accurately identify keywords and patterns in the user's command.
- Once a matching rule is found, the assistant executes the task using Python modules like os, subprocess, or **PyAutoGUI**.
- After completing the task, the assistant provides spoken feedback through an offline text-to-speech engine like **pyttsx3**.

Key Features: Fully offline, fast, lightweight, reliable, and protects user privacy.

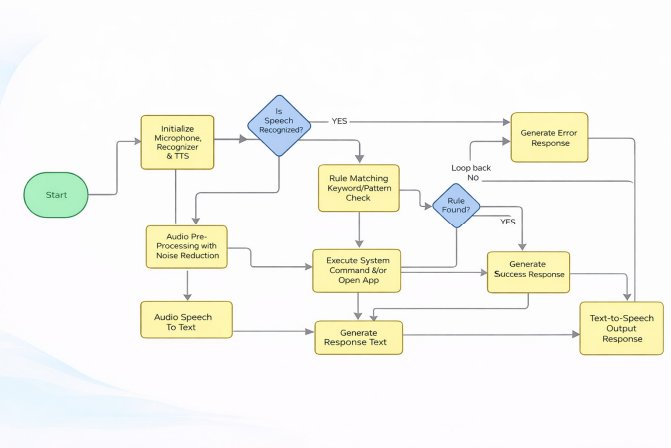

Fig. 2 Architecture of the Rule-Based Voice Assistant

8. Future Scope

- Combine rule-based logic with lightweight AI techniques for more flexibility.
- Support natural language variations and alternative phrasing.
- Context-aware command handling for multi-step tasks.
- Multilingual voice support.
- Integration with system monitoring tools for voice-controlled system status checks.
- Self-learning feature to suggest new commands based on user behavior.
- Voice-based authentication for enhanced security.
- Extend compatibility to **Linux** and **macOS**.
- Add compact offline AI models for improved intelligence while keeping data private.

9. Conclusion:

The offline, rule-based voice assistant offers a **fast, secure, and reliable** alternative to cloud-dependent assistants. Operating entirely offline is suitable for locations with poor or no connectivity. All processing is local, safeguarding user privacy and providing immediate responses to voice commands.

Lightweight, customizable, and user-friendly, the system allows users to add or modify commands as needed. While it currently handles predefined tasks, it is highly practical for **personal desktops, offline home automation, educational**

ISSN: **3107-6513**

purposes, and accessibility support. Overall, this project provides a straightforward yet effective solution for voice-controlled automation emphasizing **speed, security, and offline functionality**.

10. Results:

- System executes commands accurately and quickly.
- Fully offline operation ensures privacy.
- Lightweight design enables smooth performance on low-spec PCs.
- Limitations include accent recognition and manual command updates.
- Potential for hybrid AI integration in future to improve intelligence.

11. REFERENCES:

[1] D. Jurafsky and J. H. Martin, *Speech and Language Processing*, 3rd ed., Pearson Education, 2023.

[2] J. Allen, *Natural Language Understanding*, Benjamin/Cummings Publishing, 1995.

[3] L. Tóth and T. Haidegger, "Offline speech recognition for embedded systems," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 6, pp. 2672–2681, 2018.

[4] Alpha Cephei Inc., "Vosk Speech Recognition Toolkit – Documentation," 2024.

[5] CMU Sphinx Group, "PocketSphinx Documentation," Carnegie Mellon University.

[6] Python Software Foundation, "Python 3 Documentation."

[7] pyttsx3 Development Team, "pyttsx3: Offline Text-to-Speech Conversion Library – Documentation."

[8] Mycroft AI, "Mycroft: An Open-Source Voice Assistant Platform."

[9] Jasper Project, "Jasper: Offline Voice Control System."

[10] R. Dhanalakshmi and S. Mohan, "Design and implementation of offline voice-controlled PC assistant," *International Journal of Computer Applications*, vol. 175, no. 36, pp. 15–21, 2020.