International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 08 Issue: 12 | Dec - 2024 SJIF Rating: 8.448 ISSN: 2582-3930

SPEECH ENABLED OPERATING SYSTEM

Mr ARUL E[1] ,VISHNU PRASANTH B S[2], SUGANTH S[3] , MITHUNRAJ P[4] , KATHIRAVAN M[5]

Department of Information Technology,

Coimbatore Institute of Technology,

Coimbatore, Tamil Nadu, India.

{ [1] arul.e@cit.edu.in
[2]71762107058@cit.edu.in,
[3]71762107051@cit.edu.in,
[4]71762207204@cit.edu.in,
[5]71762107021@cit.edu.in }

ABSTRACT

The concept of the work is to create a speech interface while controlling computers using OS to manage computer operations through speech. It is more than a mere corridor is giving a smooth entry into a computer but at the same time we have made what seems to be complex operation very easy. It has been implemented on the Linux platform and uses Speech Recognition libraries with Google Speech API to convert typed direct spoken word commands to the computer fully optimized with pyttsx3 python for feedback voice output to the user. The features that are supported by this interface include typing, opening, closing, creating, deleting, renaming files, searching the web, starting other applications, etc. First it includes converting the input speech into text, the second step is parsing this text to match with the request to the coded Python codes to perform the desired action. This modularity has the Command Module for acquiring a module capturing voice, through which inputs matched will be action, the Action Module for action, and the Feedback Module through voice response confirmation. The Bash script operates with systemd- to start the interface at system boot in order to run this code under the Python virtual environment with the name myenv. This type of interface is most suitable for the disabled as it enhances convenience and flexibility in making assimilation to smart OS environments.

1 INTRODUCTION

Due to recent advances in Artificial Intelligence and interaction with the person and the computer voice interfaces are in high demand because they are natural. This project is centered on the development of an OS interface that allows the user to type, open, close, create, delete, and rename files, search the internet, and launch applications by speaking into a microphone. Since using NLP and speech recognition the system is entirely voice operated so it is very useful for disabled and might be more comfortable for the average user than the conventional inputs[6]. There are two libraries used in this project: Speech Recognition which enables the conversion of speech into text and pyttsx3 that informs by giving an audio feedback on the outcomes of commands[1]. The working flow of the system is as follows first, it detects voice, and then converts it into text by using the Google API and then maps that text file's matched equivalent Python code to perform the necessary operations. It makes the interface auto-run

when the OS comes up and Python code within a myenv virtual environment frosting the matter of usability[6]. Through the demonstration of this voice-enabled OS interface with the kernel and user applications interface the project showcases an advanced way of improving the commonly used operating system human-computer interfaces especially through vocal technology[1].

1.1 CHALLENGES

1. Speech Recognition in Noisy Environments

As one of the main users of Voice Recognition described before, noise is one of the major issues.Voice recognition is another technologic problem which mainly appears in verifying voice commands in random noise of the real environment that may worsen performance of the system[2].

2. Handling Diverse Accents and Speech Patterns

Regional accent, non fluency and clearly spoken words differences among users affect low recognition accuracy for speech in general.

3. Latency in Command Execution

Delays in processing and executing commands can lead to frustration of the users and reduction of the usability of the real time system especially if the response time is a consideration.

4. System Startup and AutoStart Issues

List of startup problems which are in compliance with the problems belonging to the genres of System Startup and AutoStart.Certain problems appear when implementing automatic start and stop of the speech-enabled OS on system boot without directly involving the user and adjusting the dependency loading and Python virtual environments on OS startup.This report examines the approach to command and control of the complexity and flexibility as aspects of war[8].

5. Managing Command Complexity and Flexibility

Designing a system in which a user can add, remove, or modify the commands without causing substantial alteration of the system is quite complex. The issue is making it flexible of command sets at the same level with the readability and further clarification of the involved processing structure.

2. RELATED WORKS

1. Advanced Speech Recognition Libraries

Libraries such as SpeechRecognition or Google APIs, which provide higher accuracy independent of noise level. Further, noise elimination algorithms can also be applied to make audio input free from noise.

2. Adaptation to Different Accents Using Machine Learning Models

Deep learning models that have been trained on multiple data sets to address multiple accents as well as speech patterns enhancing its success rates for various users.

3. Optimized Command Parsing for Lower Latency

Subprocess and asynchronous execution methods have been employed in order to respond quicker and the commands do not take time to execute as they also run at the same time.

4. Systemd Services and Shell Scripts for Autostart Management

Similar works apply systemd services and shell scripts in the autostart regulation process. They also back up

easy enabling/disabling as well as the starting and stopping of the virtual environment.

5. Customizable Command Sets for User Flexibility

Local extensions can be achieved through customization modules which are also referred to as configuration files where users can add commands freely. Some use JSON or YAML files in which users can set commands and responses; increasing versatility without compromising the codebase.

6. Environment Management with Virtual Environments or Containers

In general, to guarantee compatibility and avoid dependency issues, people use venv or Docker, which contain all the necessary libraries. This prevents rise of new problems related to setting up the system and also forms a guarantee that the system can perform effectively across other systems.

3. PROPOSED SYSTEM

The new developed HCI Speech –Enabled Operating System Interface is an improvement of Linux based operating system interface relying on NLP & Automatic Speech Processing[7]. This system permits the users to execute different operations on the terminal orally so that it is more efficient and comfortable for users. These include typing, opening or closing files, creating, deleting or renaming files, as well as simple searches on the internet and the running of applications. Also, it allows users to speak to the applications and turn the text as an input , so it fits for working with documents or programming. Through NLP, the system will be able to interpret conversational phrases which also makes the system more friendly and easy to use[3].

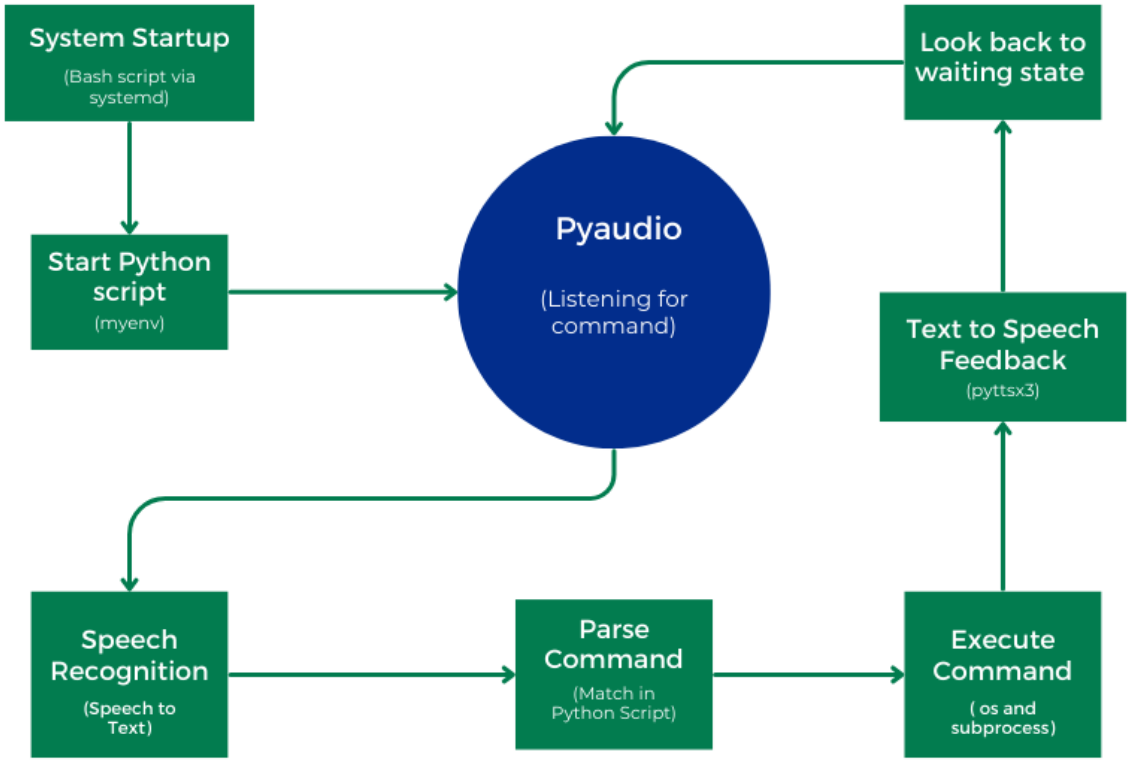
The system uses the pyttsx3 library for haptic feedback to congratylate the user for completing an instruction or for notifying the user of an error. It is easily configurable or reconstructible because users without coding or programming skills can adjust, expand or create new commands, sub-commands, tasks or functions[4]. Thus, the system is not only suitable for disabled people or people with low typing speed, but can be controlled exclusively using a voice, which excludes contact with the equipment. Use of Bash scripting scripting and ‘systemd’ makes it run the interface at system boot as an automatic startup. The digital speech input using Python code running in the virtual environment called as myenv, convert the speech into text using Google Speech API and look for the corresponding MATCH and if found then it executes the matching python code operations.

This revolutionary system changes user experience on operating systems by providing easy access, productivity, and new generation operating systems. It improves the interactivity of the system by allowing people to use their voice to interact for most of the touching tasks that can be done with a keyboard and mouse, thus making it easier for people with disabled hand movement and showing the potential of voice interface for the future of human-computer interfaces[5].

3.1 SYSTEM IMPLEMENTATION

The entire system is controlled by speech-recognition speeches operating voice system. Various technologies are used in this Application. Python acts as a major programming language. It uses SpeechRecognition for voice to text, pyttsx3 for text to speech, and pyautogui for artificial typing[4]. It has been designed solely for

Linux. With such an application installed, a voice can actually perform typing, file management, web access, and application launches. For what is said to be typed in text, determinedly with the help of Google Speech API against Python commands, it is done along with audio reinforcement of the action. Bash scripting with systemd allows starting of the service to start the script at boot and the code within the script runs in a virtual environment created as myenv in ISO format as shown below. Hence, this enables clients subscribing to this service to communicate among themselves in a more efficient, easy, and cheap manner.



3.2 ALGORITHM

Step 1: Initialize System

Import necessary libraries: Speech Recognition, pyttsx3, os, subprocess, and pyautogui.

Prepare the pyttsx3 engine for feedback in terms of speech.

Step 2: Start the System

If the operating mode includes automatic activation of virtualization, use Bash script to start the environment.

Invite the user to wait for the system to be ready to accept voice input.

Step 3: Capture Voice Input

Instantiate the Recognizer object from the Speech Recognition library.

Activate the microphone to capture voice commands.

Use Google Speech API to transcribe the audio input to text.

Inform the user in case the command is not recognized or if there is an error.

Step 4: Process the Command

Search the recognized command for keywords and phrases.

Trigger corresponding actions based on the recognized command:

Command: Open Application

If the command is "open browser," execute os.system("firefox").

If the command is "open terminal," execute os.system("gnome-terminal").

Command: Search Online

Use subprocess.run to extract the search query and open the result in the default browser.

Command: Type Text

Open a text editor, preferably gedit (os.system("gedit")), and use pyautogui to type the dictated text.

Step 5: Provide Feedback

Use pyttsx3 to give feedback for all the actions performed.

For example, after opening the terminal, say "Opening terminal"; when typing, say "Typing text"; if a command is not recognized, say "Command not recognized."

Step 6: Loop Continuously

Stay idle for the next command until the system is shut down manually.

Step 7: Manage System Startup

Use Bash scripting to run the virtual environment and then the Python script.

Set up a systemd service to ensure the program starts automatically with the operating system.

The service should allow starting or stopping as desired.

Step 8: Extend Functionality

Extend the system's functionality as needed. This may include adding new voice commands or additional features based on user needs or requirements.

Consider adding features like controlling other applications, managing system files, or integrating with external devices.

This modular approach ensures that the system is both flexible and expandable, offering a range of voice-based commands to improve the user experience with the operating system.

3.2.1 Voice-Driven System Control

The convenience of the system is that it enables users to drive their operating system exclusively by voice.

- You can for instance open terminal, browser, or any text editor by invoking a simple voice command.

For example, such operations as launching applications, working with the web, copying or renaming files, or simply typing in editors can be accomplished without user interaction.

- Open a website or type a URL by way of voice commands without having to use the keyboard or mouse so much.

  It makes work easier and fast especially in instances where one would have used many sub-menus or typed many orders. For example when the user speaks a command like “Open terminal” then the application that opens the terminal launches immediately[4].

3.2.2 Natural Language Comprehension

- It is flexible because users can command the system in their natural language

  With the help of NLP, voices can be interpreted in normal speaking language, as opposed to popular voice-activated systems that accept only specific phrases[9]. This means users do not have to remember certain phrases or how they have to be written in order to be used with the bot. For instance, instead of asking the user to search for Python tutorials, the user can say, search for me Python tutorials.

- It will analyze the executive’s specific command of a task and understand if a user used different words to pass the request to perform the task.
- Reduces the extent to which one has to master certain phrases in order to use them in a conversation making the system easy to use.

  This capability makes the system easy to use, natural and is able to span the human – computer interface, successfully.

3.2.3 Accessibility Enhancements

- Allows full user interaction with the computer through speaking to it, with many controls carried out by voice.
- Lays down voice as the main approach of interfacing, thereby minimizing the reliance of conventional input devices.

  It’s leveled to accommodate everybody within its framework thus can be extremely useful for people with physical disabilities or people who experience issues with typing. It supports the total voice control of the computer and makes voice the primary mode of interacting with the computer. All this makes a point that computing becomes open to more people so that they use it for independence and convenience.

3.2.4 Automated Task Execution

- Enables a user to perform several operations on it in a single voice command hence increasing efficiency

  Forced tasks make it possible for the system to perform all the complicated operations in one go.

- Programmatically interacts with the operations system and the applications that are involved, making high accurate automation

  They are able to start applications, navigate folders and documents, and even enter typed text from dictated constrained commands to an application.

- For instance, the command “Type ‘Hello, World!’ in the editor” would type in the text in the editor, making it easier and quicker than typing it manually.

  To accomplish its mission and achieve high-accuracy execution, this automation is driven by tools such as os, subprocess, and pyautogui.

3.2.5 Integrated Feedback Mechanism

The system provides the user with audio feedback on the commands to indicate the system’s interactions

using the pyttsx3 library. This feature reiterates-back when a command was successfully performed (e.g., “Opening browser”) or informs the client about failures (e.g., “Command not recognized”).

- Audio Feedback for Commands: Offers vocal affirmation that a specific command was carried out and with what result (for instance, ‘Opening browser.’)
- Error Notification: Notifies the users when commands corrupt or are invalid thus increasing textual communication clarity.

3.2.6 Customization and Extensibility

The two major benefits that can be derived from it are customization as well as extensibility.

- Personalized Command Options: New additions, modifications, and deletions of commands may be made depending on the users’ requirements.
- Versatility Across Applications: Flexible for use with individuals, business and organizations, or professionals.
- Future-Proof Design: Enables constant synchrony by providing for its update and alteration.

The system is versatile and can be aligned to any functional specification needed by the users.

New commands can be included, existing ones modified or others deleted which means that the system is good for personal use, for organizational purposes, or for professional use. This makes it possible for the system to continue to be useful when deployed to various systems.

3.2.7 Auto-Start Functionality

To do away with manual activation of the voice, it is set to start automatically at the time of system booting. This is done by utilizing Bash scripting and systemd service.

- System Boot Activation: Settles the voice system to begin at the system logon in order to be always prepared.
- Bash Scripting Utilization: Uses a variety of Bash scripts to initiate this process.
- Systemd Integration: Depended on systemd services for easy initialization with the Operating Systems simultaneously.
- Always Ready for Commands: Ensures system optimizes for responsiveness as soon as Windows boots up hence enhancing usability

This feature makes the system to be ever responsive to commands as soon as the computer is on, and it is precise for this reason that most operating systems are programmed this way.

3.2.8 Real-Time Typing via Voice

This is one of the finest options afforded by this project; the functionality of inputting dictated texts into applications in real-time. For instance, typing “This is a test” will open a text editor program and then type exactly what has been typed. This functionality is especially useful when, with voice commands, you can document, take notes, or even code something, which makes this feature quite specific and useful within the system.

- Instant Text Input: Makes it possible for those working with dictated texts to input this information into applications very quickly and without errors.

- Document Creation: Allows taking a note, writing a document, or coding by speaking into the virtual assistant, where a device translates the spoken words into text.
- Dynamic Program Control: Starts programs such as text editors to do text input directly without manually having to do so.
- Enhanced Productivity: Efficient in that voice commands can be used along with typing tasks on the same real-time basis

3.2.9 The Process

This system identifies speech input using Google's Speech API, converts it to text, and then parses the recognized text for matching with predefined Python functions[4]. Functions open applications, create files, search online, or type text, among other commands. To ensure starting automatically with the operating system boot once, Bash scripting is installed along with systemd integration. The complete code written in Python operates in a specific environment called myenv to ensure an easy run-through of all commands and features. A mélange of all this technology is thus incorporated towards meeting the seamless and efficient user experience[7].

3.3 CONCLUSION

However, the ability of the speech interfacing OS to give the user a feel of the OS by merely using the mouth to give the computer commands make it easier for the disabled to access the computers and also gains convenience to those with mobility in their arms and hands by reducing the amount of times they use their hands. Using speech recognition mechanism and natural language processing (NLP), the system can be able to either execute programme commands like to search documents, organise files and initiate various forms of applications, or search for information in real time feedback online using pyttsx3[6]. The system is powered on Google speech recognition API in combination with Bash scripting that can run the system on system start up. Therefore the project seeks to enhance the computing performance but at the same time make it personal and friendly.

4 REFERENCES

[1] Operating System Command Execution Using Voice Command: It is a system that allows OS command’s execution through vocal input for improving the ease and speed of its use:Paras Nath Singh,Navaneetha M,Poonam Vijay Tijare.

[2] AI-Based Desktop Voice Assistant: A behavioral desktop assistant using AI especially for performing the commands indicated by the user and intelligent work scheduling:PankajKunekar,Ajinkya,Deshmukh,Sachin Gajalwad,Aniket,Bichare,Kiran Gunjal, Shubham Hingade.

[3]Voice-Based Virtual Assistant with Security: A virtual assistant that uses safe login to perform voice directed instructions: CintamariaSimon,Rajeswari.

[4]Voice Assistant using Artificial Intelligence**:** A system that has been integrated with Artificial Intelligence so as to allow individuals easily control their devices using voice instructions;Preethi G, Abishek K,Thiruppugal S,Vishwaa D A.

[5]Artificially Intelligent Operating System with SAPI5 Voice Recognition Engine**:** An OS interface particularly with Speech application programming interface 5 for exceptional voice recognition and task performance: Sumathi S, Nivetha N, Princy Jovita J.

[6] Two-Way Speech to Sign Language Converter Application Using Python, OpenCV, and NLP**:** An interconvertible mode to translate spoken words in sign language and vice versa for a person with a disability:Shashikant Suman,Bhanu Prakash Lohani, Vijay Singh,Amar Deep Gupta,Akhilesh Kumar Khan,Anil Kumar.

[7]A Voice-Enabled Operating System for Disabled Users: An operating system developed to assist disabled persons to manage their systems using voice control;Smith et al.

[8] Speech-Based Computer Control Using CMU Sphinx: A speech recognition system for controlling computer functions using CMU Sphinx; G. Nagappan, N.Rohini

[9] Voice-Controlled Accessibility in Modern Operating Systems: Improving OS navigability by and for persons with disabilities using voice-based tools of interaction: Rodriguez and Brown

[10] Speech-Enabled Operating System Control: A Capgemini special offering in creating a smart voice control for operating the operating systems, you no longer have to type: Md. Abdul Kader;Biswajit Singha;Md. Nazrul Islam.