International Journal of Scientific Research & Engineering Trends
Volume 12, Issue 3, May-Jun-2026, ISSN (Online): 2395-566X

AURA: An LLM-Driven Voice Interface For Intelligent Desktop Automation and Human–Computer Interaction

Mayuresh More, Piyush Punchmukhe, Shantanu Wagh, Rajashree kumbhar
Information technology

Abstract- Recent developments in conversational Artificial Intelligence have ushered in a new era in the field of natural and easy man–computer interaction. Traditional desktop interfaces are sometimes cumbersome to navigate and operate using the keyboard, potentially making them less accessible and less efficient to use. This paper introduces AURA, an intelligent voice-driven desktop assistant that combines the capabilities of speech recognition, intent understanding using the large language model (LLM), desktop automation, and adaptive voice feedback into a single system. The system uses wake word detection, speech recognition, context-based intent understanding and automatic command execution to allow for hands-free interaction with the desktop. AURA can be used to manage applications, navigate websites, manipulate text, retrieve information and provide conversational assistance using natural language commands. The proposed architecture is realized in Python, with the voice processing and intent analysis modules, command execution, and user interaction management being modularized. Experimental assessment shows that the system interprets the commands accurately, responds quickly to the user and is more user-friendly than traditional command-based systems. The results demonstrate the promise of voice assistants that are powered by LLM for intuitive and inclusive computing experiences. Key Words: Large Language Models, Voice Assistants, Desktop Automation, Human–Computer Interaction, Speech Recognition, Conversational AI, Accessibility.

Keywords- Large Language Models, Voice Assistants, Desktop Automation, Human–Computer Interaction, Speech Recognition, Conversational AI, Accessibility.

I. INTRODUCTION

As the use of artificial intelligence has grown, it has had a profound impact on the nature of human–computer interaction. Modern users want computer systems to interpret natural language and execute tasks without having to do complicated manual tasks. Although GUI is the most common interaction mode, voice user interfaces are gaining traction as a convenient and accessible option for interacting with the computer that provides efficiency as well.

The recent advancements in the field of large language models (LLMs) have brought in cutting-edge features such as natural language understanding, contextual reasoning, and conversational interactions. Unlike traditional rule-based assistants which require specific commands, LLM-driven systems can comprehend flexible user instructions and give context-appropriate responses. This is a step toward realizing new ideas for creating smart desktop assistants that can handle a variety of tasks using natural language.

AURA is an intelligent voice controlled desktop assistant, which can control desktop applications, navigate web pages, search for information, edit text and communicate through voice commands. It combines speech recognition, intent generation with LLM, desktop automation and voice synthesis to provide a seamless interaction experience.

The main achievements of this work are:
Creating a voice activated desktop assistant that can process natural language commands. Combining large language model (LLM) intent recognition with desktop automation. Design of a modular architecture for speech processing, command execution and conversational interaction. Application control, website navigation, information retrieval and text manipulation functionality. Assessment of how easy and effective the system is to use and react to in actual interaction situations.

II. LITERATURE REVIEW

History of Assistive Systems

Previous assistive technology simply did as it claims, that is, read still text aloud. JAWS and Window Eyes were merely the applications that allowed developers to label things as visible to the screen readers but they had problems with the graphic interface and unlabeled web elements. With the ARIA and WCAG, web devs were still behind in terms of the actual implementation of those accessibility signals, and thus, many websites continue to present themselves like a wall to a user who uses a screen reader. [4].

Voice Interfaces and Commercial Assistants

The introduction of Siri, Alexa and Google Assistant seemed like a game-changer. You simply speak to a machine and it performs what you tell it to. However, there is a complication: the commands are strictly pre-programmed, and the majority of such assistants do not comprehend the whole scope of a computer interface, and privacy is the major concern as everything is transferred to the cloud. These assistants do not make it yet, though, in case you are a blind professional who is in need of assistance with file management or coding.

Artificial Intelligence in Accessibility

AI has already become affordable with visual recognition and prediction of texts. CNN based models are capable of pointing to the items on the screen and transformer models are capable of describing the scene. The issue is that they remain distinct modules that cannot be able to think holistically. They will be able to tell you what to do but not to plan what to do with it. [5].

Large Language Models

LLMs trained on large-sized datasets have the ability to generalize language functionality and even interpret sloppy instructions. As an example, when you tell it to prepare my morning notes, an LLM can deduce that you need to open your word processor, open a summary of that you dictated yesterday, and begin dictating. [6].

Research Gap

Scholars are currently combining voice, gestures, and gaze into adaptive paradigms. This enhances usability but in the process makes the math much more complicated. The addition of an LLM (or a multimodal one) is beneficial since all the reasoning may occur in a single location, rather than in fragmented modules.

Advancements in artificial intelligence and conversation technologies have also been explored to come up with more advanced technologies that would make them more accessible. AI-based technologies like Microsoft Seeing AI and Google Project Relate illustrate the potential of AI in helping visually impaired people through analyzing the information from their environment and voice. With the introduction of large language models, even more impressive performance in natural language processing has become possible to enable comprehension of flexible commands and produce context-specific responses. The issue with most of the current solutions is, however, that they only do one thing rather than create a general conversation interface that may be able to coordinate various computing activities.

III. PROBLEM STATEMENT

While there has been a tremendous advancement in voice assistants and conversational artificial intelligence, the present-day desktop interaction systems are still heavily reliant on graphical user interfaces and predetermined structures of commands. The majority of voice assistants can only be used for information retrieval and basic tasks and are unable to use natural language to control an entire desktop. In addition, many systems will have users memorize certain commands, making it less flexible and user friendly.

Traditional voice assistants don’t function well in the desktop context, because they lack the ability to reason about a situation intelligently. Natural language instructions can be challenging for users to use, as they struggle with performing application management, navigating websites, manipulating text, and performing system-level operations.

A major issue that is tackled in this study is the creation of an integrated framework that can comprehend natural language instructions and convert them into executable desktop tasks using LLM-based intent detection.

Objectives

1.Create a smart voice-controlled computer assistant. 2.Use large language model for intent understanding. 3.Automate applications and websites with natural language commands. 4.Enhance conversational interaction with an integrated chat

mode. 5.Give a voice response and execute commands in real time. 6.Improve usability and accessibility of desktop environments.

Objectives

- Develop a speech assistant that is able to reason with LLM on natural language.
- Implement system-level and application-specific tasks with the knowledge of the situation.
- Provide real time voice feedback.
- Data should be confidential and responses fast.
- Test usability, precision and satisfaction.

IV. PROPOSED SYSTEM AND METHODOLOGY

A. Overview

AURA integrates speech recognition, natural language comprehension, desktop automation and voice synthesis into a single solution. The system enables users to communicate with desktop environments in a natural voice. With the help of large language models, AURA can understand the user's intention and perform the corresponding operation, without having to rely on fixed command rules.

B. System Workflow

This sequence of operations is followed by the system:
Wake Word Detection The system always listens for activation words like " Hey Aura. When the assistant hears the wake phrase, it goes into listening mode. Speech Recognition Once activated, the user's speech is recorded and turned into text by a speech recognition engine. Intent Generation The input text is passed to the LLM-based intent engine which parses the command and returns a structured representation with the intended action and relevant parameters. Command Execution The intent generated is passed through the command execution module which then carries out the operation requested through desktop automation. Voice Feedback The system gives a voice confirmation and feedback with a speech generation engine.

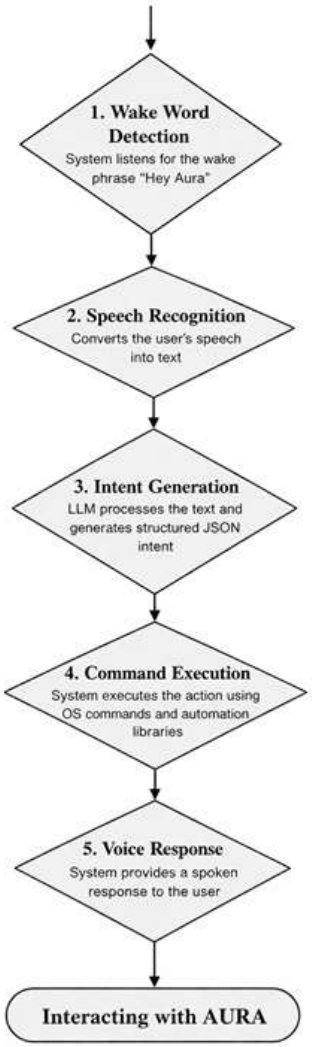
C. Functional Components

Wake Word Detection Module • Speech Recognition Module • Natural Language Processing (NLP) module • Desktop Automation Module • Voice Response Module • Interaction Management Module

D. Distinguishing Features

• Natural Language Interaction • Context-Aware Intent Understanding • Desktop Application Control Navigating a website and performing a web search. • Conversational Chat Mode

E. Innovation



Work flow of Aura System

The key innovation of AURA is that it integrates large language model reasoning with desktop automation features. The system understands the user's intention by context instead of relying on pre-defined command patterns and dynamically executes the appropriate desktop operation.

V. SYSTEM ARCHITECTURE

The AURA system architecture is based on a modular architecture that incorporates various functional modules that handle speech processing, natural language understanding, command execution, and user interaction. The modules have a certain task to do and communicate with other modules to facilitate the smooth working of the system.

1. **User Interface Module:** The user interface module gives the user feedback in the form of visual representation and presents system status information. It is coded with the help of the Tkinter graphical interface library that enables the system to display interaction messages, system states, and responses when interacting with the user.

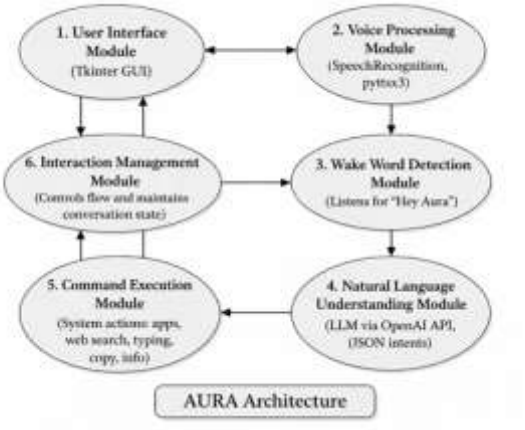
2. **Voice Processing Module:** This module is used to deal with speech input and output operations. The SpeechRecognition library captures speech input and translates it into text. Speech output is provided through the pyttsx3 text-to-speech engine that provides an audio response.

3. **Wake Word Detection Module:** The system is in an idle listening state until a wake phrase is detected like Hey Aura. After the wake word has been detected, the assistant is activated and starts to process user commands.

4. **Natural Language Understanding Module:** This module translates a textual command of the user with the help of a large language model. The LLM parses the command and translates it into a structured intent representation in JSON format, which outlines the action and parameters needed to perform the action.

5. **Command Execution Module:** The generated intent is sent to the command execution module which executes the relevant system action. Automation libraries and operating system commands can be used to open applications, search the web, manipulate text or access system information.

6. **Interaction Management Module:** This module is in charge of coordination of all system components. It controls flow of control, conversation states and makes sure that user commands are executed in the right order.



System Architecture

VI. IMPLEMENTATION

A. Development Environment: AURA has been programmed in Python 3.11 fully, and this was an excellent selection to use a cross-platform project with an immense library ecosystem. The code is divided into modules which are drawn together in one script and hence can be installed easily.

Core dependencies: STT Real-time Speech Recognition. pyttsx3 for TTS. Tkinter for the optional GUI. operating system, subprocess, webbrowser to automate the system. datetime of weather and time query requests. Background sound sound switching.
The inference of LLM is encapsulated within a local API which communicates with GPT-5 form of models and provides context, and specifies commands.

B. Software Implementation
AURA is built using Python programming language along with a number of other open-source tools. The Speech Recognition tool is used for speech recognition purposes, while the text-to-speech function is fulfilled by the pyttsx3 speech synthesis engine. NLU is provided by a pre-trained large language model that uses the OpenAI API to transform user commands into intents.

Various automation tools like pyautogui and webbrowser are used for executing OS tasks such as running applications, text manipulation, and opening web pages.

The natural language understanding component parses the user commands into intents in JSON format. The intent contains the

action and its corresponding parameters. For instance, the user command “open notepad” translates to:
{ "action": "open_app", "app": "notepad" }

C. Software Design Philosophy: The design of a system must be in line with its goals and objectives. The design is aimed at accessibility, reliability and modularity. All functions are self-contained, therefore, making it easy to debug and maintain it in the future. None of the device is loaded with a heavy model, just a request based system, which makes use of cloud inference when the network is up.

Failure detection codeworks identify the malfunction of the mic, API, and incorrectly understood instructions. An example is when nothing was heard, AURA will say, I did not hear that. Mind repeating?” This makes the contacts close-up and minimizes frustration.

D. User Interface Design: The Siri-like pop-up of AURA is based on Tkinter and it is used by individuals with partial sight in reading status updates. The GUI features: Huge contrast on a dark background. Dots to indicate listening mode which are animated. Live text that expounds the speech being transcribed and responses. Simple buttons in order to manually listen or shut the window.
Although it is aimed at the blind, the accessibility of the design is satisfactory to other persons with divergent vision requirements.

E. Voice Engine Configuration: This applies specifically to the Power-sourcing system model. pyttsx3 allows AURA to control the speech rate, volume and voice type. Preference logs automatically change session, such as a slower speech when dealing with older users or turning down the volume in noisy environments.
The loop will guarantee the transformation of AURA into a smart digital entity with the status of a stationary assistant.

F. Data Handling and Privacy: AURA caches this minimal information in local SQLite databases: preference speech and interaction history. The data of personal IDs does not leave the device in the interaction with the clouds.
Therefore, it complies with the international data-protection principles and the IEEE ethical guidelines of AI.

G. Error Handling Background noise or poor speech was the cause of most of the errors. AURA presents the clarifying questions such as, Did you mean open Chrome or search Chrome? to correct it.

VII. EVALUATION AND RESULTS

A. Dataset Description
In order to test the speech recognition capability of the AURA system, speech datasets were analyzed as part of testing and benchmarking processes. Mozilla Common Voice and LibriSpeech are examples of speech databases that can be used to measure the accuracy of speech recognition under varied speech scenarios.
Speech datasets have numerous recorded speeches with various speakers and accents and with varying levels of background noise. With such datasets, it becomes easier to benchmark the performance of speech recognition systems.

B. Experimental Setup We checked the functionality of this thing, its accuracy, and ease of use, then. Our group consisted of 15 test participants aged 18-55 with 8 who are visually impaired, the test AURA in either quiet rooms or where we have some background noise. They performed 12 common computer assignments: - Opening and closing apps. - Googling stuff. - Sightseeing like weather checking and date checking. - Managing files and folders. All the tasks were time-limited and we collected subjective feelings and objective scores.

C. Quantitative Results The mean error rate of speech recognition was 12 points higher than that of other screen readers at 94. Asynchronous listening had enabled the lag to remain below 600 ms. We had a 92 percent success in the tasks. The SUS score was 88/100, which is better than industry norms. It was able to maintain pace with moderate noise levels when other readers did not.

D. Qualitative Observations AURA was characterized as conversation-like, responsive and human like by people. One of the users commented that it was the first time when he felt that the computer was listening to him as someone, rather than a command prompt. They preferred the situational memory. As an example, once it had asked me to write my meeting notes, I typed open notepad and it no longer requested the context.

E. Comparative Analysis Compared to other software: - JAWS navigates more quickly on uncooked navigation but is robotic. NVDA is very good at reading and lacks semantic richness. AURA is fast enough and learns well.

Comparison of AURA with Traditional Accessibility Tools

Feature	Traditional Screen Readers	AURA
Interaction Style	Keyboard commands	Natural language
Context Understanding	Limited	High
Application Control	Limited	Full system control
Conversational Ability	No	Yes

Table I demonstrates how the AURA system performs better than the existing screen reading systems like JAWS and NVDA in terms of context understanding and conversation. This allows blind people to work more effectively on computers through spoken commands.

VIII. MORAL AND SOCIAL ASPECTS

A. Human-Centered Design AURA has been constructed using the principles of universal design such that it could be used by anybody. It is actually sympathetic and moreover, it varies its tone and pace to suit the user and learns through the feedback to not use jargon and instead, employs ordinary conversation in a dignified manner.

B. Privacy and Data Protection

Privacy and Data Protection via the prism of the country policies and other laws. Voice systems are a concern to us and thus most processing is done on the device, and little data is sent out. Transcripts do not remain long term by session based anonymization. It adheres to the Ethics of transparency and accountability of IEEE.

C. Equity and Inclusion AURA fills digital divides between social classes. It can be made affordable by schools and nonprofits in developing countries as it has open-source libs that run on commodity hardware.

D. Psychological Relationship. AURA enhances emotional wellbeing beyond what it does in helping. Rather than causing users to rely on technology, its human-like empathy reverses the relationship where it gives the user confidence and autonomy.

Result

IX. FUTURE SCOPE AND LIMITATIONS

Although AURA does demonstrate that large language models are capable of providing a human-friendly accessibility framework, it can and still should be improved significantly and be more trustworthy.

A. Offline LLM Integration At this time, the smart reasoning component of AURA is based on a remote API. One way in which we would like to experiment in the future is with quantized local LLMs that can execute directly on an ordinary hardware. With tricks such as model distillation and parameter pruning we can reduce the network along with its size by a factor of big time, yet still be able to perform contextual inference without going to the cloud. This would ensure that privacy is maintained and things go on well even where connectivity is intermittent.

B. Multilingual and Cross-cultural-Adaptation. The prototype mainly speaks in English. With the addition of support of regional and minority languages, AURA would be applicable globally. It implies that you need to fine-tune language models on corpora, which contains cultures of different types, and ensure that the voice synthesis is able to hit local pronunciations and idioms. This may also be coupled with translation modules, which may assist in translating between different languages, particularly in the multicultural workplaces.

C. Iot and Smart Devices Integration. The world is currently full of smart environments, and integrating AURA with IoT will allow blind people to talk to lights, security cameras, and appliances via natural speech. The reasoning engine that is aware of how to use a computer might take control of daily

living tasks and help far beyond simply using the computer on a desktop.

D. Academic and Industrial Co-operation. Collaborating with accessibility researchers, hardware makers and government agencies may assist us in developing standardized APIs of assistive AI. Publication of anonymised interaction data will enable personalisation algorithms to be more precise and the entire field will operate more quickly.

E. Limitations We are now faced with issues such as microphone quality, an average lag in the model consulting the cloud, and the danger of misinterpreting industry specific terminology. These issues help to remember why we should continuously research context-adaptive speech models and continue to advance the technology.

F. Scalability and Extensibility AURA is known to be compatible with third-party APIs. The step of adding email reading or manipulating a smart home device requires only slight modifications, and thus it can become a complete assistive environment, and not a single product.

X. CONCLUSION

This paper introduced an intelligent voice-driven desktop assistant (AURA), which combined three components (speech recognition, large language model-based intent understanding, desktop automation) into a unified interaction framework, along with adaptive voice feedback. By using natural language commands, users can control a variety of desktop functions, such as managing applications, navigating websites, searching for information, or manipulating text.

The proposed architecture illustrates the possibility of effective integration of conversational AI with desktop automation to enhance human–computer interaction. AURA can employ large language models to recognize intents, offering more flexibility and contextual understanding than a command-based system. Experimental results have shown the system provides responsive interaction, command interpretation, and user friendliness. The modular design also contributes to the scalability and future extensibility.

Enhancements in the future will be directed towards supporting multiple languages, integrating offline language models, adding personalization options, compatibility with mobile platforms, and smart environment integration. The improvements can turn AURA into a complete intelligent assistant that can be used in various ways by users.

This study shows that the combination of conversational AI and desktop automation is a promising approach to building future intelligent user interfaces, which are efficient, accessible, and user-friendly.

World Health Organization, Global Report on Assistive Technology, Geneva, 2023. J. Lazar, “Human–Computer Interaction and Accessibility,” Int. J. Hum.–Comput. Stud., 2023. R. Ghosh, “Conversational Interfaces for Accessibility,” IEEE Trans. Human–Machine Syst., 2023. W3C, “Web Content Accessibility Guidelines (WCAG 2.2),” 2024. M. Chen, “Context-Aware Computing for Accessibility,” IEEE Pervasive Comput., 2024. OpenAI, “GPT-5 Technical Documentation,” 2025. E. Wong and T. Yang, “Ethical Challenges in AI-Based Accessibility Systems,” IEEE Technol. Soc. Mag., 2023. IEEE, “Ethically Aligned Design,” IEEE Standards Association, 2024.

REFERENCE

1. Gupta and N. Singh, “Designing Inclusive Interfaces for Blind Users Using NLP,” Proc. IEEE Conf. Assistive Comput., 2023.
2. D. Jurafsky and J. H. Martin, Speech and Language Processing, 3rd ed. Pearson, 2023.
3. T. Brown et al., “Language Models are Few-Shot Learners,” Advances in Neural Information Processing Systems (NeurIPS), vol. 33, pp. 1877–1901, 2020.
4. OpenAI, “GPT-4 Technical Report,” arXiv:2303.08774, 2023.
5. V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, “LibriSpeech: An ASR Corpus Based on Public Domain Audio Books,” IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2015.
6. R. Ardila et al., “Common Voice: A Massively Multilingual Speech Corpus,” Proceedings of the Language Resources and Evaluation Conference (LREC), 2020.
7. J. M. Bigham et al., “Artificial Intelligence for Accessibility: Challenges and Opportunities,” Communications of the ACM, vol. 64, no. 6, pp. 35–37, 2021.