Advanced International Journal for Research (AIJFR)
E-ISSN: 3048-7641 ● Website: www.aijfr.com ● Email: editor@aijfr.com

A Python-Based AI Virtual Assistant for Task Automation

Ankita S. Nathe[1], Vaishanavi Biradar[2], PriyankaLanjewar[3], Ananta Gole[4], Dhanshri Lanjewar[5], Nikhil Gade[6]

[1]Assistant Professor of Information Technology Department
KaviKulguru Institute of Technology & Science (KITS), Ramtek. Nagpur,India.
[2,3,4,5,6]Department of Information Technology KaviKulguru Institute of Technology & Science (KITS), Ramtek. Nagpur,India.

Abstract
Recent advances in Artificial Intelligence have enabled the widespread adoption of voice-based virtual assistants for desktop automation and human–computer interaction. However, most existing assistants rely heavily on continuous internet connectivity, raising concerns related to latency, privacy, and reliability in low-connectivity environments. This paper presents **Cosmo**, a Python-based AI virtual assistant that employs an **adaptive hybrid architecture** capable of dynamically switching between online and offline processing modes. In online mode, Cosmo leverages large language models (LLMs) through cloud APIs for complex conversational tasks, while in offline mode it utilizes lightweight local speech recognition and text-to-speech engines to ensure uninterrupted functionality. Experimental evaluation demonstrates that the proposed hybrid approach improves response reliability and reduces average task latency during network failures while maintaining acceptable accuracy. The results indicate that Cosmo provides a practical, privacy-aware solution for desktop automation, education, and productivity applications.

Keywords—AI Virtual Assistant, Hybrid Architecture, Offline Speech Recognition, Desktop Automation, Human–Computer Interaction, Privacy-Aware Systems

1. Introduction

The Voice-based AI assistants have become an integral component of modern human–computer interaction, enabling users to interact with systems through natural language. Popular assistants such as Alexa, Google Assistant, and Cortana primarily rely on cloud-based processing, which provides high accuracy but introduces challenges related to privacy, latency, and dependency on stable internet connectivity. These limitations restrict their usability in offline or resource-constrained environments. To address these challenges, this paper proposes Cosmo, an adaptive hybrid AI virtual assistant designed to operate efficiently in both online and offline modes. The system dynamically detects network availability and selects appropriate processing pipelines accordingly. The main objective of this research is to investigate whether a hybrid online–offline architecture can enhance reliability and usability without significantly degrading performance.

A. Related Work

Previous research on virtual assistants has explored cloud-based conversational agents, multimodal interaction systems, and desktop automation tools. Joshi et al. proposed a personal desktop assistant focusing on task automation but relied entirely on online services. Bohouta and Kepuska compared speech recognition APIs, highlighting trade-offs between accuracy and computational cost. Recent studies have emphasized usability and privacy concerns in voice assistants, suggesting the need for offline-capable solutions. Despite these contributions, limited research has focused on adaptive hybrid architectures that seamlessly integrate online intelligence with offline reliability. Cosmo addresses this gap by combining cloud-based LLMs with local speech processing engines.

2. System Architecture

B. Core architectural components

First, The architecture for a Python-based AI virtual assistant hybrid online/offline functionality consists of a core, modular system that switches its capabilities and data sources based on internet connectivity. The assistant is composed of several independent components, including a user interface, input processors, a command interpreter, and a task execution module.

C. Input layer

This module captures user requests and converts them into a format system can understand. Speech Recognition: Uses Python libraries like Speech Recognition to transcribe audio from the user's microphone into text. Online Mode: Utilizes more powerful, cloud-based APIs like Google's Speech Recognition for higher accuracy. Offline Mode: Uses local, lightweight models such as Vosk or Whisper, which require no internet connection. Text Input: Receives typed commands directly from the user through the assistant's graphical user interface (GUI).

D. Processing layer

This is the central "brain" of the assistant, interpreting the user's intent and deciding on the appropriate action. Natural Language Processing (NLP): Intent Recognition: Analyses the processed text to determine the user's goal, such as "open an application" or "search for information". Online Mode: Can integrate with larger, more powerful LLMs via APIs for complex, conversational tasks. These models are trained on vast datasets and provide more nuanced responses. Offline Mode: Relies on a local knowledge base with pre-defined commands or simpler NLP techniques like keyword matching and regex extraction. Data and Knowledge Management. Online Mode: Retrieves real-time information from external APIs for services like Wikipedia, weather forecasts, or news updates.

E. Task execution layer

This module performs the actions based on the command decision. Desktop Automation: Uses Python libraries to interact with the local machine. It stimulates mouse and keyboard actions to control GUI elements. OS and sub processes modules execute system commands like opening applications or managing files. Hybrid Operation: The assistant automatically selects the correct functionality based on the current mode. For example, a search command would use a web API in online mode but switch to the local knowledge base or a simple "cannot connect" response in offline mode.

F. Output layer

This module converts the assistant's response into a format the user can understand. Text-to-Speech (TTS): Uses libraries like pyttsx3 to convert text responses into spoken audio. Online assistants may use more advanced, natural-sounding voice APIs. Graphical User Interface (GUI): Displays text, visual elements, and the results of a command directly on the desktop screen.

G. Hybrid functionality workflow

The hybrid capability is implemented through conditional logic that checks for an active internet connection. Start: The assistant is always listening for a "wake word" using a lightweight, locally processed model. Check Connectivity: When a command is received, the system first checks if it can access the

Internet. If online: The request is sent to the cloud for advanced processing, like complex LLM queries or real-time data retrieval from web APIs. If offline: The request is handled locally by the on-device NLP model, which matches it against a pre-defined set of system commands and offline data. Execute and Respond: The appropriate task is executed (e.g., open a local app or perform a web search), and the response is delivered via TTS or the GUI.

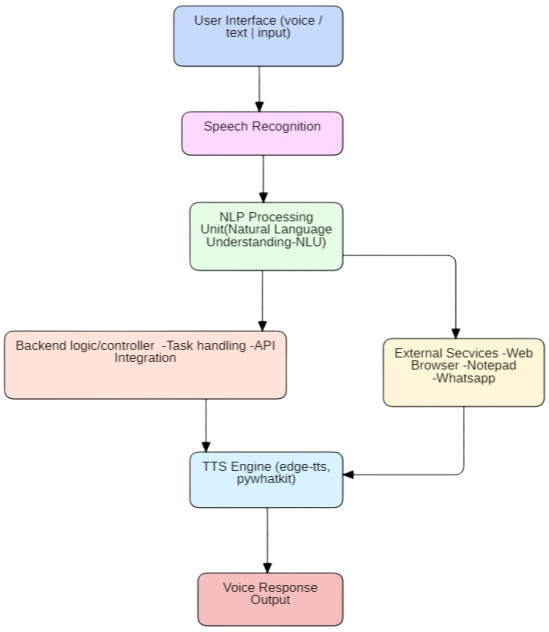

Fig.System Architecture

PROPOSED SYSYTEM

We are proposing a system in an efficient way of implementing a Personal voice assistant, Speech Recognition library has many in-built functions, that will let the assistant understand the command given by user and the response will be sent back to user in voice, with Text to Speech functions. When assistant captures the voice command given by user, the under lying algorithms will convert the voice into text. And according to the keywords present in the text.

This is made possible with the functions present in different libraries. Also, the assistant was able to achieve all the functionalities with help of some API's. We had used these APIs for functionalities like performing calculations, extracting news from web sources, and for telling the weather. We will be sending a request, and through the API, we're getting the respective output. API's are very helpful in performing

things like calculations, making small web searches. And for getting the data from web. In this way, we are able to extract news from the web sources, and send them as input to a function for further purposes. Also, we have libraries like Random and many other libraries, each corresponding to a different technology.

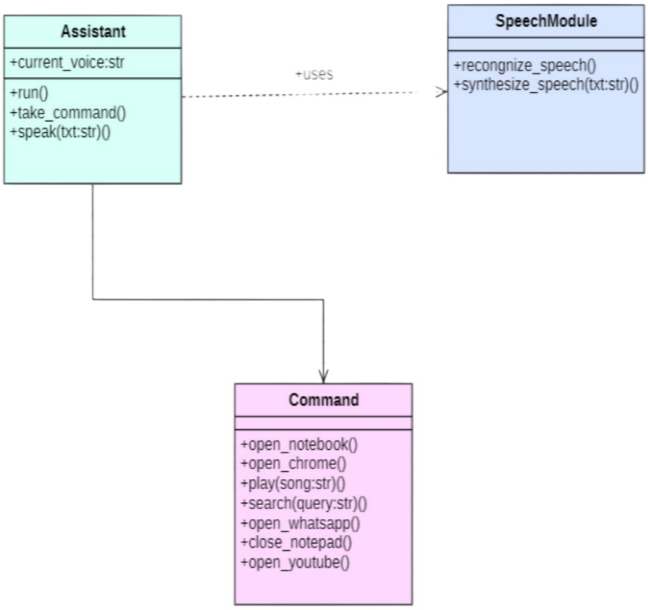

Fig.UML Diagram

RESULT

The Cosmo AI Virtual Assistant was evaluated based on its functionality, accuracy, responsiveness, usability, and offline capability. Each module of the assistant—voice recognition, natural language processing, task execution, text-to-speech, and GUI—was tested under different environments, including both online and offline conditions. The goal was to validate whether Cosmo could act as a dependable, real-time, AI-powered assistant capable of automating system tasks and engaging in intelligent conversation. The results demonstrate Cosmo's effectiveness, its strengths, and areas where future improvements can be made.

Fig.Home Interface

The real-time system statistics panel serves not just as a monitoring tool but also as a performance diagnostic module. It reflects how much CPU, RAM, and disk space Cosmo is consuming, which is vital

during continuous use, especially in embedded or resource-constrained systems. Similarly, the command log panel on the right serves a dual purpose: it helps users track their spoken commands and also assists developers in tracing module responses, errors, or system prompts.

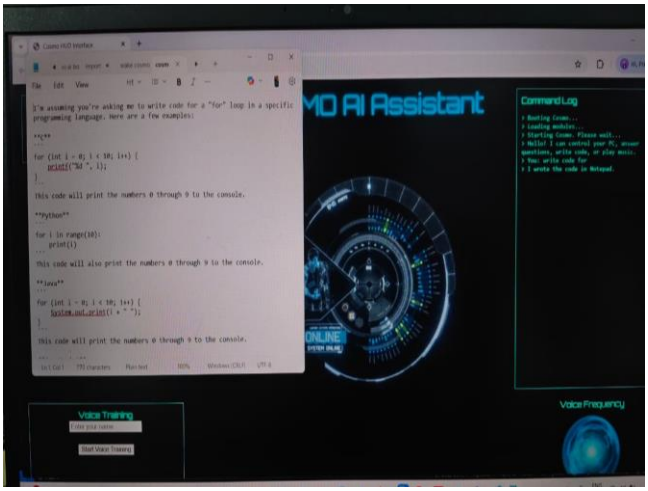

Fig. Opening applications by voice

The image shows a scenario where the user has given the AI assistant a voice/text command to “open Notepad”, and the assistant successfully executed the command by launching Notepad on the desktop. The Notepad window is open on the left side of the screen, containing examples of for loops in C, Python, and Java, each printing numbers 0 through 9. On the right side of the screen, within the “Command Log” section of the futuristic Cosmo HUD Interface, the assistant’s activity is displayed in green terminal-style text

ACKNOWLEDGMENT

The authors express their sincere gratitude to the Information Technology Department at Kavikulguru Institute of Technology & Science (KITS), Ramtek, for providing the necessary infrastructure and support to conduct this research on python based virtual assistant we are thankful to our colleagues and research advisors for their invaluable guidance and constructive feedback throughout the project. Special appreciation is extended to the IEEE and arXiv communities for their extensive resources, which enriched our literature review.

REFERENCES

1. Rabin Joshi, Supriyo Kar, Abenezer Wondimu Bamud and Mahesh T R, (2023). Personal A.I. Desktop Assistant, 2(2), 54-60 International Journal of Information Technology, Research and Applications (IJITRA)
2. Geraldine Shirley.N.: Virtual Control hand Gesture Recognition System using Raspberry Pi, ARPN Journal of Engineering and Applied Sciences, vol. 10, no. 7, pp. 2989-2993, (2015).
3. S. Arora, K. Batra, and S. Singh. Dialogue System: A Brief Review. Punjab Technical University.

4. R. Mead. 2017.Semio: Developing a Cloud-based Platform for Multimodal Conversational AI in Social Robotics. 2017 IEEE International Conference on Consumer Electronics (ICCE).
5. K. Noda, H. Arie, Y. Suga,and T. Ogata. 2014. Multimodal integrationlearning of robot behavior using deep neural networks. Elsevier: Robotics and Autonomous Systems.
6. R. Pieraccini, K. Dayanidhi, J. Bloom, J. Dahan, M.l Phillips. 2003. AMultimodal Conversational Interface for a Concept Vehicle. Eurospeech2003.
7. G. Bohouta and V. Z Kepuska. 2017. Comparing Speech Recognition Systems (Microsoft API, Google API And CMU Sphinx). Int. Journal of Engineering Research and Application 2017.
8. Keerthi Premkumar, K Gerard Joe Nigel, -Smart Phone Based Robotic Arm Control Using Raspberry Pi, Android and Wi-Fi‖, 2nd International Conference on Innovations in Information Embedded and Communication Systems, (2015).omic variables as common pervasive risk factors and the empirical content of the Arbitrage Pricing Theory. Journal of Empirical finance, 5(3): 221–240.
9. F. L. I. Dutsinma, D. Pal, S. Funilkul, and J. H. Chan, ‘‘A systematic review of voice assistant usability: An ISO 9241–11 approach,’’ Social Netw. Comput. Sci., vol. 3, no. 4, p. 267, Jul. 2022, doi: 10.1007/s42979-022-01172-3.
10. J.W.Anderson,‘‘Sternberg’s triangular theory of love,’in Encyclopaedias of Family Studies. Hoboken, NJ, USA: Wiley, 2016, pp. 1–3, doi:10.1002/9781119085621.wbefs058.
11. T. Mettler, M. Sprenger, and R. Winter, ‘‘Service robots in hospitals: New perspectives on niche evolution and technology affordances,’’ Eur. J. Inf.Syst., vol. 26, no. 5, pp. 451–468, Sep. 2017, doi: 10.1057/s41303-017-0046-1.
12. B. A. Carroll and A. C. Ahuvia, ‘‘Some antecedents and outcomes of brand love,’’ Marketing Lett., vol. 17, no. 2, pp. 79–89, Apr. 2006, doi:10.1007/s11002-006-4219-2.
13. S. Schuetz and V. Venkatesh, ‘‘‘Research perspectives: The rise of human machines: How cognitive computing systems challenge assumptions of user-system interaction,’’’ J. Assoc. Inf. Syst., vol. 21, no. 2, pp. 460–482,2020, doi: 10.17705/1jais.00608.
14. I. Seeber, E. Bittner, R. O. Briggs, T. de Vreede, G.-J. de Vreede, A. Elkins,R. Maier, A. B. Merz, S. Oeste-Reiß, N. Randrup, G. Schwabe, and M. Söllner, ‘‘Machines as teammates: A research agenda on AI in team collaboration,’’ Inf. Manage., vol. 57, no. 2, Mar. 2020, Art. no. 103174,doi: 10.1016/j.im.2019.103174.
15. S. A. Rijsdijk, E. J. Hultink, and A. Diamantopoulos, ‘‘Product intelligence: Its conceptualization, measurement and impact on consumer satisfactions’J. Acad. Marketing Sci., vol. 35, no. 3, pp. 340–356, Sep. 2007 doi: 10.1007/s11747-007-0040-6.
16. R. Frischknecht, ‘‘A social cognition perspective on autonomous technology,’’ Compute. Hum. Behav., vol. 122, Sep. 2021, Art. no. 106815, doi:10.1016/j.chb.2021.106815.