Revolutionizing Interaction: A Desktop Voice Assistant Powered by Artificial Intelligence

Siddhish Nirgude
Department of Mechanical Engineering
Dr. Vishwanath Karad MIT World Peace University
Pune, India
nirgudesiddhish@gmail.com

Vandana Jagtap
Department of Computer Engineering and Technology
Dr. Vishwanath Karad MIT World Peace University
Pune, India
vandana.jagtap@mitwpu.edu.in

Pallavi Parlewar
Department of Electronics and Communication Engineering,
Shri Ramdeobaba college of engineering and management
Nagpur Maharashtra, India
parlewarpk@rknec.edu

***Abstract*—AI has completely changed the way humans interact with technology, ushering in a new era of human- computer interaction. This study combines state-of-the-art methods for speech synthesis and recognition to design and deploy an AI-powered desktop voice assistant. The technology uses sophisticated algorithms to translate spoken words into text (Speech to Text) and text to speech (Text to Speech) smoothly. This allows users and computers to communicate in a natural and intuitive way.**

The desktop voice assistant, which can comprehend user orders and respond intelligently, is a virtual companion that was created using the Python programming language. The system's overall efficiency and user experience are improved by the accurate interpretation of spoken words made possible by the incorporation of Artificial Speech Recognition. Speaking commands can be used to carry out tasks, obtain information, and manage desktop apps, improving accessibility and efficiency.

This study highlights the useful applications of AI in everyday computing in addition to showcasing its technological capability. This desktop voice assistant with AI capabilities provides a window into interactive computing's future, when natural language processing and artificial intelligence will combine to improve and simplify our digital lives. It is a significant advancement toward human-machine interfaces that are easier for users to interact with.

Keywords—Text-to-speech, Speech-to-text, Desktop voice assistant, Python, Virtual assistant, Artificial speech recognition.

I. Introduction

AI is presented as a means of enabling computers to think and make decisions similar to humans, with the study of AI focusing on human-like idea generation, learning, decision-making, and problem-solving. The primary objective of AI research is to enhance computer capabilities related to human knowledge processing, such as reasoning and problem- solving, while encompassing intangible elements like perception and linguistic intelligence [1,5,6].

The text underscores the exponential growth of data generated by both humans and machines, highlighting the limitations of human ability to process vast amounts of information effectively. It emphasizes that AI, coupled with deep learning models, is poised to become the cornerstone of future business decision-making [2,8,18].

Furthermore, the text highlights the increasing integration of voice assistants into everyday life, with multinational companies adopting them to enhance user interactions with Machines [9,15,17,19]. These virtual assistants are described as particularly beneficial for various user groups, including seniors, the visually impaired, children, and disabled individuals, due to their ease of use and voice- based interactions.

The specific focus of the text is on a Desktop Voice Assistant with Artificial Speech Intelligence, designed to process voice input and provide text-based output, catering to a range of tasks, from web searches to setting alarms. The system is described as being developed for Windows users using Python programming and leveraging machine learning for training.

In conclusion, the text introduces the evolution of AI, the rising importance of virtual assistants, and the development of a desktop-based voice assistant as a practical application of AI technology [3,14,16,20]. It emphasizes the potential for AI and deep learning to reshape business decision- making processes and highlights the accessibility and versatility of voice-based virtual assistants in enhancing human-machine interactions.

II. Literature Review

Voice assistants have a long-standing history, with their origins dating back to 1962 when IBM introduced the shoebox IBM, a device capable of recognizing spoken digits and processing up to 16 words [10]. These assistants, including notable ones like Cortana developed by Microsoft for desktop use, rely on language processing to execute tasks. The fundamental purpose shared by all voice assistants is voice configuration processing, a breakthrough made possible by modernistic technology such as Artificial Intelligence (AI) [10].

In the realm of voice assistants, notable figures like Sutar Shekhar and various researchers have collaborated to develop apps that function based on voice commands.

These advancements have led to features like sending messages through voice commands, particularly designed to assist individuals who are partially sighted. There are ongoing efforts to create engines capable of recognizing local languages, promising increased accessibility and usability [11].

Python programming language, renowned for its power and ease of use, plays a significant role in these developments. The integration of speech recognition and the Pyttsx3 module ensures that the software operates seamlessly and accurately, simplifying the user experience [4,7,13].

Additionally, the software has specific features tailored for partially sighted users, such as repeating commands to confirm their accuracy. Moreover, the software is designed to continuously listen and process user demands until the interactions are complete, ensuring a user-friendly and responsive experience [12]. This continuous evolution invoice assistant technology showcases the ongoing efforts to enhance accessibility and user interaction through innovative applications and features.

III. PROPOSED METHODOLOGY

Human speech is widely acknowledged to be crucial to our daily interactions in both our personal and professional life, and Speech-to-Text has many uses. Sounds and noises make up basic audio data. Speech in humans is an exception to that rule. For Speech-to-Text tasks, the available training sets consist of:

Input features: audio snippets of spoken speech

Target labels: a transcript of those specific words in text.

- Data Pre-Processing

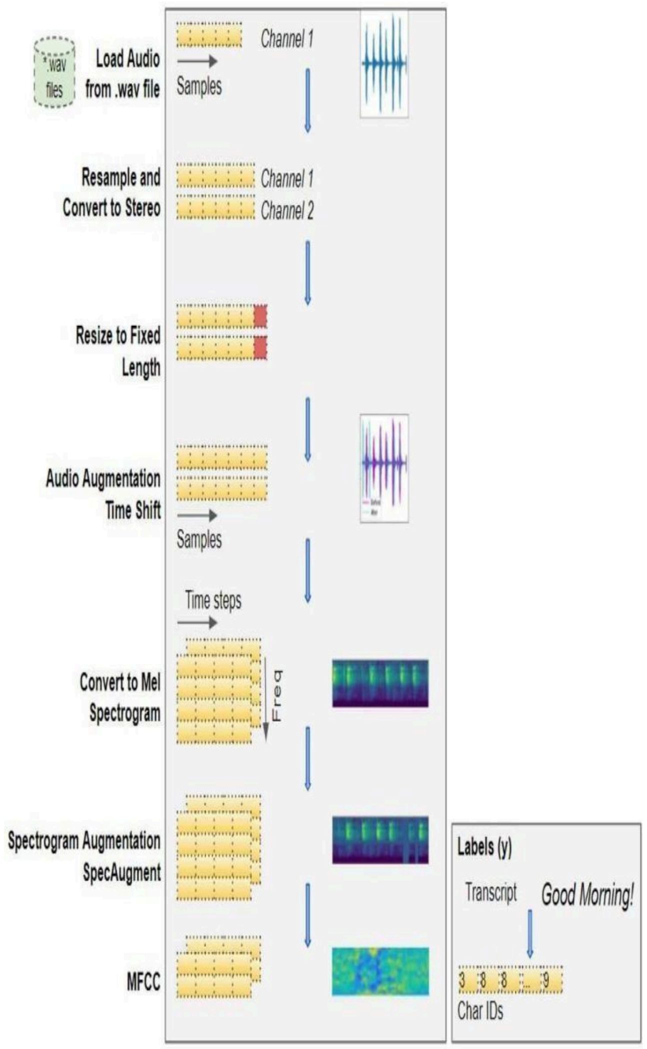
.wav files
Load Audio from .wav file
Samples
Channel 1
Resample and Convert to Stereo
Channel 1
Channel 2
Resize to Fixed Length
Audio Augmentation Time Shift
Samples
Time steps
Convert to Mel Spectrogram
Freq
Spectrogram Augmentation SpecAugment
MFCC
Labels (y)
Transcript
Good Morning!
3 8 8 ... 9
Char IDs

Fig. 1 Load Auto Files.

- Load Audio Files

A 2D Numpy array is loaded with audio files in ".wav" or ".mp3" formats. Each row of the array represents a channel (mono or stereo), and the columns show amplitude measurements over time. Two channels make up a 3D array in stereo audio. Reading files, decoding them, and adjusting for sampling rate are all part of loading.

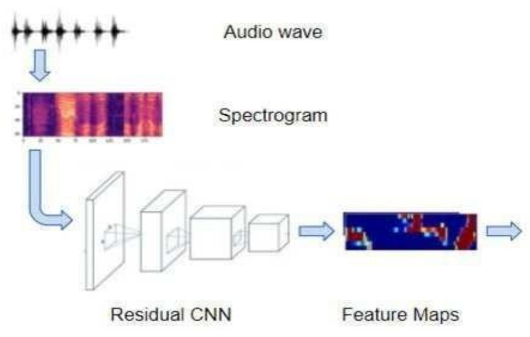

Fig. 2 A convolutional network processes spectrograms to create feature maps.

- Preprocessing of Data

To ensure uniformity in audio data for utilization in deep learning models, it is imperative to standardize dimensions concerning sample rate, channels, and duration. This process encompasses resampling audio to a consistent rate, standardizing the number of channels, and adjusting duration through padding or truncation techniques. Additionally, noise removal algorithms may be warranted to enhance audio quality.

- Data Augmentation

The augmentation of data involves using techniques such as time shifting, pitch and speed variations. This introduces diversity in the input data, which helps the model's generalization capability.

- Spectrogram Representation

The conversion of raw audio into Mel Spectrograms serves to encapsulate the audio characteristics in image form by decomposing the frequencies present.

- Representation of MFCC

To improve processing efficiency for speech-related activities, Mel Spectrograms can be transformed into Mel Frequency Cepstral Coefficients (MFCCs), which provide a simplified representation of important frequency coefficients.

- Spectrogram Augmentation

Employing SpecAugment, which entails the application of Frequency and Time Masking, enables the augmentation of Mel Spectrogram images, consequently bolstering the robustness of the model against variations.

- Preparing the Target Label

The transcript text is transformed into character IDs to make the process of training the model easier. This gets the target labels ready for alignment with the input features.

- Architecture

A conventional convolutional network with a few residual CNN layers that processes input spectrogram images and returns feature maps of those images.

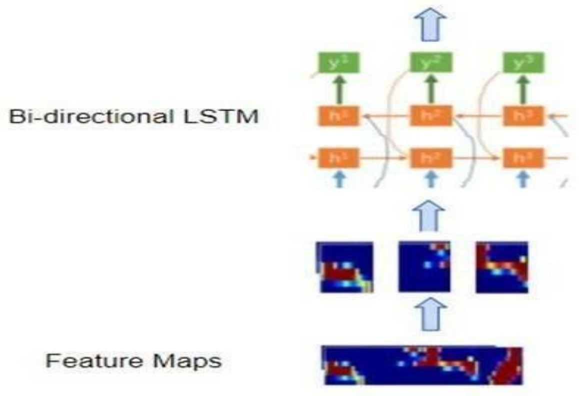

Fig. 3 Recurrent Network Process framesfrom the feature map.

A conventional recurrent network comprises several Bidirectional Long Short-Term Memory (LSTM) layers, which analyze feature maps as a sequence of discrete timesteps or 'frames' aligned with the desired output character sequence. LSTMs, commonly employed in recurrent layers, are adept at capturing long-term dependencies in sequential data. Thus, the network transforms continuous audio feature maps into a discrete representation, facilitating subsequent processing.

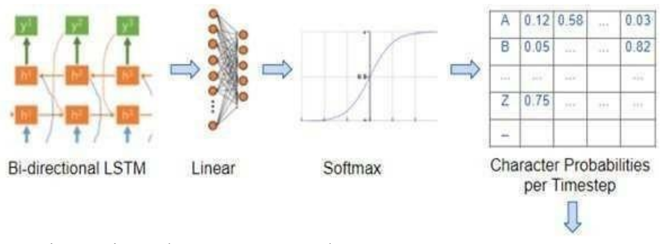

Fig. 4 Linear layer generates character probabilities for each timestamp.

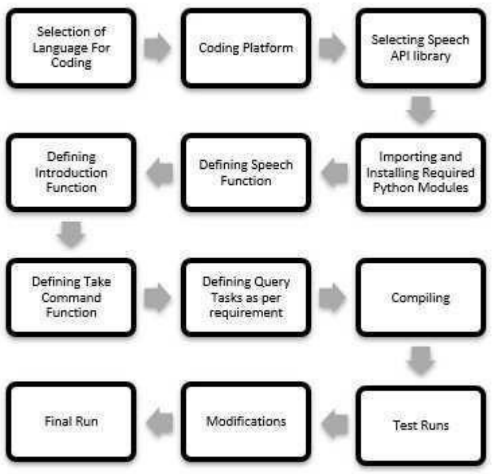
Selection of Language For Coding
Coding Platform
Selecting Speech API library
Importing and Installing Required Python Modules
Defining Speech Function
Defining Introduction Function
Defining Take Command Function
Defining Query Tasks as per requirement
Compiling
Test Runs
Modifications
Final Run

Fig. 6 Step-by-Step Methodology

In addressing the challenge of aligning audio frames with characters in the transcript, the lack of pre-segmented audio data poses a significant hurdle.

Variability in character durations, gaps, mergers, and repetitions further complicates alignment. To overcome this, the Connectionist Temporal Classification (CTC) algorithm is employed, providing a solution to the intricate task of aligning CTC Algorithm:

In situations where the input is continuous and the output is discrete, and where it is not possible to use distinct element boundaries to verify the input to individual output sequence segments, the CTC is employed to align the input and output sequences.

CTC works in two modes:

CTC Loss (while learning): It attempts to train the network to increase the likelihood of producing the accurate transcript using a ground truth target transcript.

CTC Decoding (during Inference): Since there isn't a final transcript available, we must infer the most likely character order.

A step by step approach to implementing thisproject was used as follows

A. Selection of programming language:

There are various options when it comes to selecting a computer language to program in viz. C, C++, Java, Python, etc. After reviewing options, python was selected due to its simple syntax and greater integration with AI Based Modules.

B. Coding Platform:

Once Programming Language was decided, Coding platform for writing, Compiling and running the code was to be decided. Here, VS studio Code by Microsoft was selected keeping in mind the available modules and Speech Library integration.

C. Selecting Speech API Library:

The Assistant requires to speak multiple sentences as a part of Interactive Conversation to demonstrate AI Capabilities. For such a Speech API library is needed which contains all the required modules and pre-defined functions for speech capabilities.
There are various options pertaining to Speech API with different data sets and voice characteristics comparison of which are given further ahead.
After researching and shortlisting multiple APIs based on various parameters, SAPI 5 (Speech API 5.0 by Microsoft) was selected due to its inbuilt module integration with VS studio Code and its large and varied Database for speech capacities.

D. Importing and installing required modules:

VS Studio Code has some inbuilt and some add-on modules for command routines pre-defined. Some of which are used in this project viz.pyttsx3(SAPI5 -Python integration),Wikipedia, Speech Recognition, OS, Date time etc. Some of these modules were inbuilt while some needed to be installed using pip command. Further, on these modules were imported in the beginning of code.

E. Defining Speech Function:

A speech function is needed to be defined for setting assistant speech characteristics like voice type, Volume, Etc.

F. Defining Introduction Function:

To enhance Interactive experience of user, theAssistant needs some Introductory Statements whenever user boots it up. For such an introduction function is developed which wishes the user according to time of day and with basic conversation.

Defining Takecommand Function:

The project implements a command recognition system where the Assistant listens to user commands through the microphone. A "take command" function utilizes a Speech Recognition module, such as Google Speech Recognition, to process speech input into machine language. Users can specify the desired natural language for speech input, such as English (India).

G. Defining Query Tasks:

The assistant executes various predefined tasks based on user-defined speech input. Tasks are processed into queries for imported modules, employing ladder if-else loops. Multiple modules cater to diverse tasks, defined with standard syntax as needed.

H. Compiling:

The Code is compiled to check for errors and compatibility issues using inbuilt python libraries.

I. Test Runs:

Multiple Test runs are conducted to check for bugs or glitches. Modifications required are identified.

J. Modifications:

Based on feedback from Testing phase, appropriate modifications and/or enhancements are added to the code.

K. Final Run:

The modified code is run for final stage to identifyand rectify any errors if found.
This concludes the methodology for project implementation.

Results and Discussions

The Assistant exhibits commendable accuracy and efficiency in executing various tasks, ranging from reading Wikipedia summaries to opening applications, playing music, and providing the date and time. Its modular code structure allows for customization, enabling users to adapt it to diversetasks. Moreover, the potential for enhancing its capabilities by incorporating machine learning algorithms is recognized, suggesting a promising avenue for future development.

This research delves into the design and implementation of digital assistance, leveraging open- source libraries and Python programming. The project's versatility is highlighted as additional features can be seamlessly integrated without disrupting the existing system, leading to a reduction in manual workload. Notably, the assistant extends its functionality beyond simple commands, accommodating user queries related to opening applications and performing other operations.

Looking forward, voice assistants seem to have bright futures ahead of them. As these assistants continue to advance, more complex functions will become possible to handle with voice commands, including appointment scheduling, ticket booking, and audio/video playback controls. To put it simply, voice assistants have the potential to become indispensable elements of our everyday existence, streamlining intricate activities and revolutionizing our interactions with technology.

REFERENCES

[1] Leon Reicherts, Yvonne Rogers, Licia Capra, Ethan Wood, Tu Dinh Duong, and Neil Sebire. 2022. It's Good to Talk: A Comparison of Using Voice Versus Screen-Based Interactions for Agent-Assisted Tasks. ACM Trans. Comput.-Hum. Interact. 29, 3, Article 25 (June 2022), 41 pages. https://doi.org/10.1145/3484221

[2] Olujimi, P.A., Ade-Ibijola, A. NLP techniques for automating responses to customer queries: a systematic review. Discov Artif Intell 3, 20 (2023). https://doi.org/10.1007/s44163-023-00065-5

[3] Dwivedi, Y. K., Kshetri, N., Hughes, L., Slade, E. L., Jeyaraj, A., Kar, A. K., ... & Wright, R. (2023). "So what if ChatGPT wrote it?" Multidisciplinary perspectives on opportunities, challenges and implications of generative conversational AI for research, practice and policy. International Journal of Information Management, 71, 102642.

[4] Kumar, A., Kaur, D., & Pathak, A. K. (2022, October). VOICE ASSISTANT USING PYTHON. In 2022 International Conference on Cyber Resilience (ICCR) (pp. 1-4). IEEE.

[5] Joshi, A., & Akram, S. V. (2022, November). Personalized Desktop App Based Interactive Means of Zira Voice Assistant. In 2022 International Interdisciplinary Humanitarian Conference for Sustainability (IIHC) (pp. 1071-1077). IEEE.

[6] Kumar, S., Gupta, V., Sagar, S., & Singh, S. K. (2023). IoT based Personal Voice Assistant. arXiv preprint arXiv:2305.17788.

[7] Umapathi, N., Karthick, G., Venkateswaran, N., Jegadeesan, R., & Srinivas, D. DESKTOP'S VIRTUAL ASSISTANT USING PYTHON.

[8] Pandey, A., Vashist, V., Tiwari, P., Sikka, S., & Makkar, P. (2020). Smart voice based virtual personal assistants with artificial intelligence. Artificial Computational Research Society, 1(3).

[9] Saibaba, C. M., Waris, S. F., Raju, S. H., Sarma, V. S. R. K., Jadala, V. C., & Prasad, C. (2021, August). Intelligent voice assistant by using OpenCV approach. In 2021 second international conference on electronics and sustainable communication systems (ICESC) (pp. 1586- 1593). IEEE.

[10] Sorte, B. W., Joshi, P. P., & Jagtap, V. (2015). Use of artificial intelligence in software development life cycle—a state of the art review. International Journal of Advanced Engineering and Global Technology, 3(3), 398-403.

[11] Jagtap, V. S. (2013). KarishmaPawar.(2013)"Analysis of different approaches to Sentence-Level Sentiment Classification". International Journal of Scientific Engineering and Technology, 2(3), 164-170.

[12] Dandwate, P., Shahane, C., Jagtap, V., & Karande, S. C. (2023). Comparative study of Transformer and LSTM Network with attention mechanism on Image Captioning. arXiv preprint arXiv:2303.02648.

[13] Hebbar, D., & Jagtap, V. (2022). A Comparison of Audio Preprocessing Techniques and Deep Learning Algorithms for Raga Recognition. arXiv preprint arXiv:2212.05335.

[14] Kunekar, P., Deshmukh, A., Gajalwad, S., Bichare, A., Gunjal, K., & Hingade, S. (2023, January). AI-based Desktop Voice Assistant. In 2023 5th Biennial International Conference on Nascent Technologies in Engineering (ICNTE) (pp. 1-4). IEEE.

[15] Akash, S., Jayaram, N., & Jesudoss, A. (2022, May). Desktop based Smart Voice Assistant using Python Language Integrated with Arduino. In 2022 6th International Conference on Intelligent Computing and Control Systems (ICICCS) (pp. 374-379). IEEE.

[16] Reddy, P. S. V., Kalki, T. P., Roshini, P., & Navaneethan, S. (2023, January). Varoka-Chatbot: An Artificial Intelligence Based Desktop Partner. In 2023 International Conference on Artificial Intelligence and Knowledge Discovery in Concurrent Engineering (ICECONF) (pp. 1-6). IEEE.

[17] Faiz, Z., Srivastava, V., & Khoje, S. (2022). Virtual voice assistant for smart devices. ECS Transactions, 107(1), 4315.

[18] Barnwal, V. K., Shaw, A., Sarkar, K., Chakraborty, S., & Mukhopadhyay, A. K. (2023, June). ARIVA: Artificial Intelligence Enabled Voice Assistance System using Natural Language Processing.In 2023 8th International Conference on Communication and Electronics Systems (ICCES) (pp. 769-776). IEEE.

[19] Nag, T., Ghosh, J., Mukherjee, M., Basak, S., & Chakraborty, S. (2022). A Python-Based Virtual AI Assistant. In Emerging Technologies in Data Mining and Information Security: Proceedings of IEMIS 2022, Volume 2 (pp. 585-594). Singapore: Springer Nature Singapore.

[20] Mahesh, T. R. (2023). Personal AI Desktop Assistant. International Journal of Information Technology, Research and Applications, 2(2), 54-60.