JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Zeus: The Personal Voice Assistant

Mrs. P. Navya [1], Ashish Gajjela [2], Borra Meghana Chowdary [3], Arya Shreya [4]
Assistant Professor [1], Student/Research Scholar [2], Student/Research Scholar [3], Student/Research Scholar [4]
Department of Artificial Intelligence and Machine Learning,
Vignana Bharathi Institute of Technology, Hyderabad, India

Abstract: This research paper seeks to create a voice assistant that leverages ChatGPT, a substantial language model, to respond to user inquiries. Users can ask questions in a conversational approach because of the voice assistant's user-friendly interface and ability to understand natural language queries. The system functions by turning the user's voice input into text, which the ChatGPT then analyses to produce pertinent responses. The voice assistant then communicates with the user by translating these responses back into speech. The system uses machine learning techniques to progressively enhance its performance, allowing it to offer precise and trustworthy responses to a variety of queries. The suggested voice assistant can be further improved to cater to particular domains, such as education, entertainment, and healthcare, and has the potential to enhance the user's experience with voice-based interfaces.

IndexTerms - **Voice Assistant, Natural Language Processing, Natural Language Understanding, OpenAI API, pvporcupine, pvrecorder, speech-recognition, pyttsx3, Jetson Nano Board**

I. INTRODUCTION

Due to its convenience and user-friendliness, voice assistants have experienced a significant increase in popularity in recent years. These systems enable voice instructions, doing away with the requirement for manual input, to be used to carry out activities. However, traditional voice assistants frequently have trouble deciphering natural language questions and giving precise answers. A new breed of voice assistants has evolved to overcome this constraint, utilizing natural language processing (NLP) and machine learning techniques to provide more complex responses. This research study suggests creating a sophisticated voice assistant that uses Natural Language, a powerful language model, to give precise answers to user inquiries. Enhancing user interactions with voice-based interfaces to promote more natural and conversational interactions is the main goal.

This paper describes the features of contemporary voice assistants in this context and focuses on their benefits, such as enhanced user experiences, precise responses, and multitasking skills. It also covers the drawbacks, such as dependence on the internet, privacy issues, and pronunciation difficulties. The problem statement highlights the current shortcomings of conventional voice assistants in comprehending and answering customer inquiries, resulting in less-than-ideal user experiences. By enabling more natural user interaction with the voice assistant, the suggested solution seeks to alleviate this constraint and promote a more seamless and user-friendly experience.

The core objective of this research project is to develop a voice assistant that employs the capabilities of ChatGPT, a powerful language model, to provide accurate and contextually relevant responses to user queries. The project aims to improve the overall user experience with voice-based interfaces by enabling users to interact conversationally and receive dependable responses. Additionally, the project seeks to highlight the potential advantages of leveraging natural language processing and machine learning algorithms to continually enhance the system's performance. Through customization for specific domains, such as education or entertainment, the voice assistant can offer tailored responses to cater to diverse user needs. Ultimately, the paper envisions the development of a versatile voice assistant that serves as an effective virtual personal assistant, enabling users to perform tasks efficiently through voice interactions.

This paper is structured as follows: Section II provides details on the literature survey. Section III explains the methodology. Section IV shows how to implement the methodology in Section III. Section V shows the experimental results of the research project.

II. LITERATURE SURVEY

In the modern world, when machines are replacing people in many tasks, we train our machines to think like humans and do tasks independently. Based on this circumstance, the idea of a voice assistant emerges, capable of performing a variety of tasks for people based just on their speech. The virtual assistant can filter out certain user commands and return information that is relevant to the command [1].

The system proposed by Nivedita Singh, Dr. Diwakar Yagyasen, Mr. Surya Vikram Singh, Gaurav Kumar, and Harshit Agrawal has a range of functionalities, such as the ability to set individual listening time frames for commands, repeat user input requests, and a choice of male or female voices. It provides several capabilities, including music playback, email, and text management, searching Wikipedia, starting up system programs, and online browsing. By allowing flexible listening times and ongoing prompting for precise information retrieval, the system prioritizes user preferences first. For a more customized experience, users can choose between male and female speech options [1].

Tulshan explained that because of continuous typing, there may be the possibility of injuries to the fingers of the user. To avoid such problems, we need to design a system in which we can get our work done through our voice commands. The voice commands will be recognized by the system and the recognized words will be synthesized if they are appropriate or makes some sense then that will be printed on the screen and after this again by recognizing the specific keywords the program will be compiled and executed [2].

The steps of the system created by Abhay Dekate, Chaitanya Kulkarni, and Rohan Killedar include data collecting in the form of speech, voice analysis and text conversion, data storage and processing, and generating speech from the output of the processed text. Speech data is collected in the first phase and saved as input for processing in the second phase. The input speech is repeatedly processed and translated to text using STT throughout the second phase. The converted text is then examined and analyzed by a Python script to determine the appropriate action to take in response to the instruction. Finally, output is produced via straightforward text-to-speech conversion when the response has been recognized [3].

The HC3 (Human ChatGPT Comparison Corpus) dataset, which includes over 40,000 question-answer pairs with both human-generated and ChatGPT replies, was first published by Biyang Guo and other authors. The study uses this dataset to conduct a variety of investigations, including content detection trials, linguistic analysis, and human assessments. The results of linguistic analysis and human assessments provide insight into the subtle differences between human and ChatGPT interactions, which in turn shapes thoughts on the future development of Language Model Systems (LLMs). Additionally, the tests aimed at identifying ChatGPT-generated material provide important insights that might be useful recommendations for the development of technologies intended to recognize AIGC [4].

According to A. M. J. Hashana, P. Brundha, and M. U. Ahamed Ayoobkhan, deep learning is an application of machine learning that makes use of multi-layer neural networks to learn from data and make predictions. One well-known example is ChatGPT, which makes use of attention mechanisms to concentrate on important input pieces and comprehend large amounts of text material. ChatGPT develops the capacity to understand contextual nuance in natural language through training on large text datasets, including articles and books. With the aid of this deep learning methodology, ChatGPT can identify patterns and correlations in the input text, enabling the creation of coherent and contextually appropriate responses like those of human conversation. This method is useful for a variety of natural language processing applications, including question-answering, text summarization, and language translation. Even though deep learning in ChatGPT is beneficial, it's crucial to realize that it has drawbacks, including complexity, data requirements, and computational demands during training [5].

III. METHODOLOGY

Each Personal Voice Assistant will have its wake word, such as "Alexa" for Amazon's Alexa or "Hey Google, Hi Google" for Google Assistant. A wake word is a phrase or trigger that causes a Voice Assistant to turn on. It lets the device know that the user wishes to interact with it or begin following commands. They are essential for Voice Assistants since they continuously listen to the sounds and wait till the wake word is said. Wake words are used to limit mistaken activations and ensure that the voice assistant only responds to human commands.

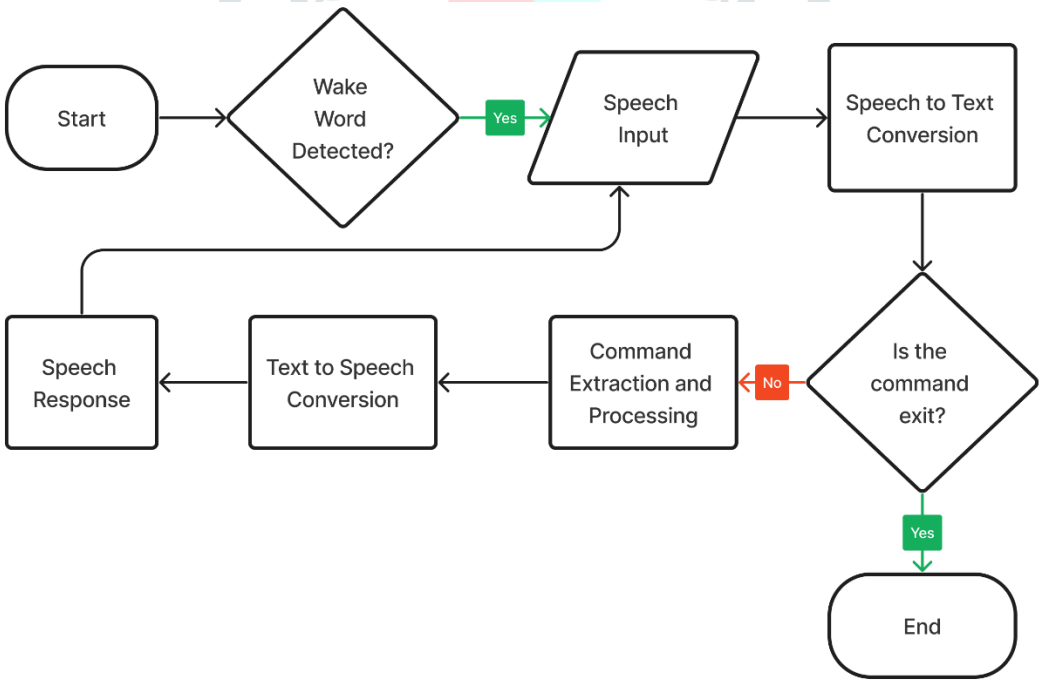

Fig. 1. Methodology

Wake word detection is the standard first step for any voice assistant, thus ours does too. The system will wait for the wake word as soon as it powers on and will keep waiting until it is spelled out. The system will begin listening to the speech as soon as it recognizes the wake word. It will convert the speech to text after listening to it. It will verify whether the command is "exit" after converting to text. If "exit" occurs, the system will be terminated. If not, the command will be extracted and processed.

It will first determine if the command is requesting information about itself at the command extraction and processing step. If it does, the system will reply with information about itself. Otherwise, it will determine whether the command is a task. If it is a task, it will carry out the task and report that it has been completed. If not, it will conclude that the command is a question that the user needs to have a response to. As a result, the system will ask ChatGPT a query and then wait for a reply. The command extraction

and processing stage will be finished after ChatGPT delivers the response, at which point the system will return the best response it has received from ChatGPT.

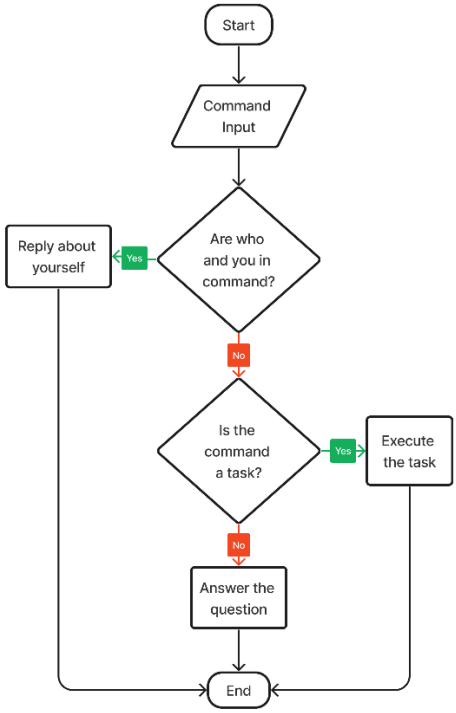

Fig. 2. Command Extraction and Processing

The system will translate the response into speech after receiving it from the earlier stage and will then communicate it to the user. The system continues to wait for the next command. This cycle will continue until the user enters the command "exit," at that point the system will stop looking for commands and wait until the wake word is detected once more.

IV. IMPLEMENTATION

Our voice assistant begins with the wake word detection stage, as we covered in Section III. The pvporcupine model from Picovoice, a business that specializes in voice recognition and natural language processing methods, will be used for this step. In the Picovoice console, we can train our own words for wake word detection. We must download the model after training it. Depending on the platform we are building for, the downloaded file will be in ppn format, and we can utilize it with the PvPorcupine SDK.

Since I'm using Python, I'll use pip to install and use the pvporcupine library. The Picovoice firm has created another module called pvrecoder, which will start the microphone and wait until it receives the wake word, allowing the pvporcupine module to capture input continually. The system will either proceed to the next phase or wait until it discovers the wake word if the wake word is found.

The Recognizer object from the speech-recognition module, which will convert the speech into text, and the pyttsx3 engine from the pyttsx3 module, which will turn the text into voice, will both be initialized when the wake word has been detected. The engine object will now be used to greet the user, and the recognizer object will be used to begin recording the command or instruction. voice will be inputted into the Recognizer object, which will turn the voice into text.

The command needs to be processed once it has been turned into text. The system will notify the user and shut down if the first step in processing the command is to determine whether the text is "exit". If the text is not "exit" it will then be examined to see if it contains a command, instruction, or query, such as "What is react?".

If there's a command, it will let the user know that it's being carried out. After that, it will carry out the command and notify the user that it has done so. If it's not a request or directive, it's a question. So, utilizing the OpenAI API, we will now ask the question to OpenAI's gpt-3.5-turbo. It will reply with a text response to the query we sent. With the help of the pyttsx3 engine object, we will now speak this text to the user.

The system will wait for a new command or instruction once it has answered the query or completed the command's execution. Up until the computer recognizes "exit" as the command, this loop will be repeated. It will notify the user and shut down itself after receiving the command "exit".

For the implementation described above, we mostly used the Python libraries Tkinter, speech-recognition, pyttsx3, pvporcupine, pvrecorder, openai, and os. The Tkinter library is used to give the user a way to interact with the system that allows them to copy the system's responses to their inquiries as well as the questions they have asked. The speech-recognition library is used to translate user-provided voice input into text. The text responses and any instructions the system wants to give the user are converted into speech using the pyttsx3 module. Wake word detection is carried out using the pvporcupine and pvrecorder

libraries. To ask queries of ChatGPT and receive answers from ChatGPT, the user uses the openai library. The user's instructions, such as "open Chrome," are carried out by the operating system library.

V. EXPERIMENTAL RESULTS

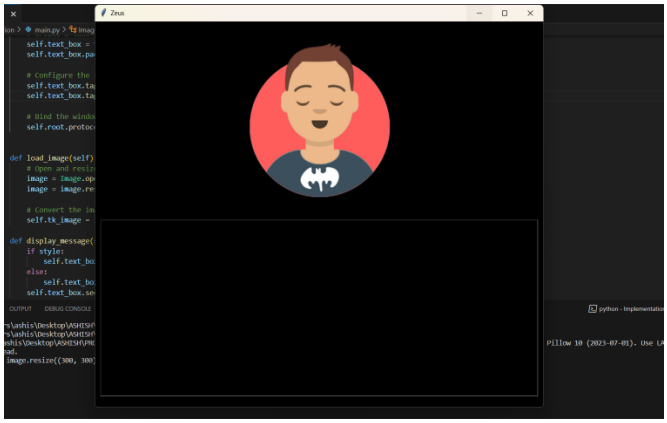

Fig. 3. Waiting for Wake Word

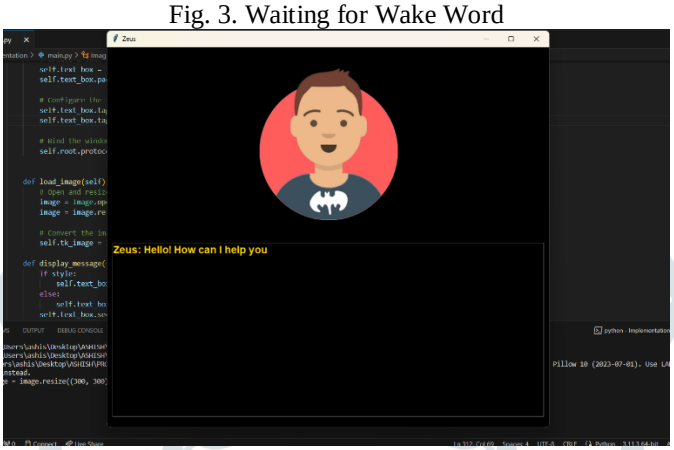

Fig. 4. Wake Word Detected

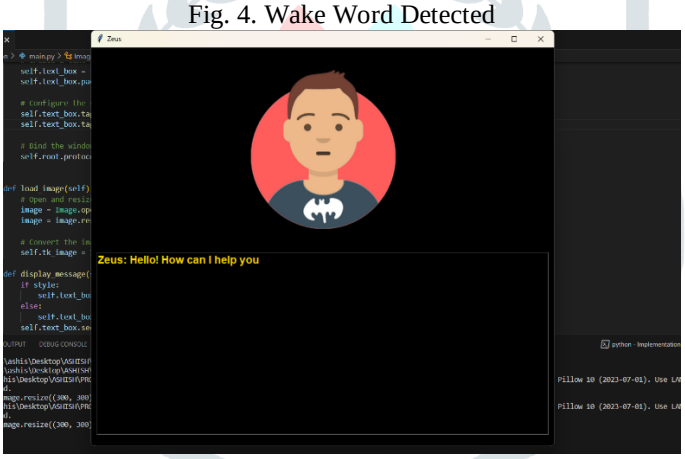

Fig. 5. Listening to User Input

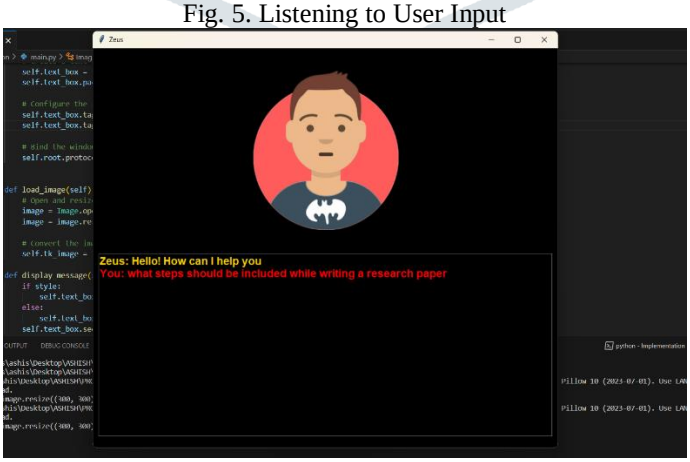

Fig. 6. Processing the Input

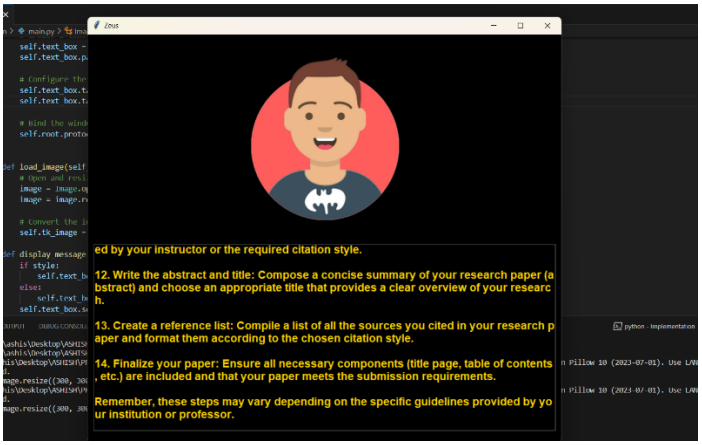

Fig. 7. Telling the Response to the User

Fig. 8. Executing Task

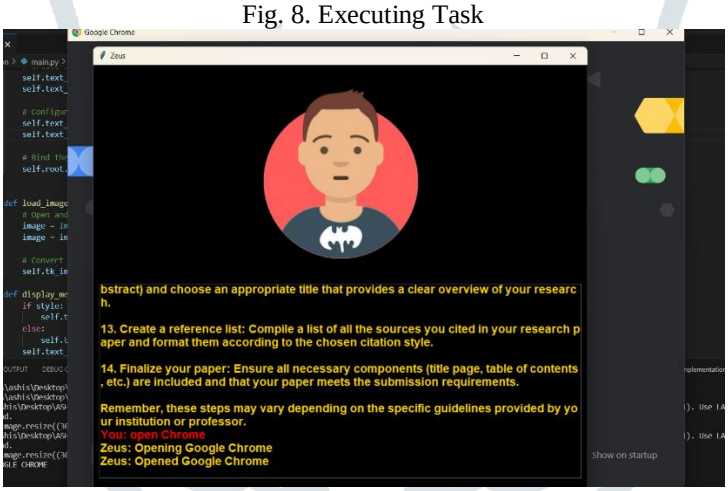

Fig. 9. Executed Task

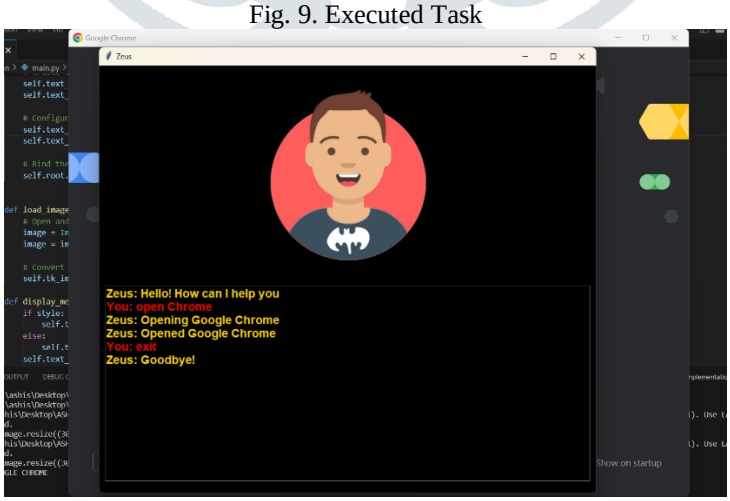

Fig. 10. Exiting the System

VI. Conclusion

In conclusion, our research paper shows a comprehensive framework for developing a Personal Voice Assistant using ChatGPT. It also tells the significance of a well-defined wake word for user engagement. The systematic wake word detection, speech-to-text conversion, and command processing, coupled with the integration of diverse Python libraries, create a robust and user-friendly interface. Noteworthy is the dynamic interaction with ChatGPT, facilitated by the openai library, allowing the voice assistant to intelligently respond to user queries beyond predefined commands. The system's versatility in executing user instructions through the operating system library adds practical utility. Our work contributes to the evolution of voice assistants, offering a personalized, efficient, and innovative solution that sets the stage for further advancements in natural language processing and voice-activated technologies.

Acknowledgment

We would like to thank Mrs. P. Navya for her valuable comments and suggestions to improve the quality of the paper. We are also grateful to Dr. K. Shirisha Reddy for helping us review our work regularly. We would also like to thank the Department of Computer Science Engineering (AI and ML), VBIT Hyderabad for providing us Jetson Nano Boards and all the help we needed.

References

[1] Harshit Agrawal, Nivedita Singh, Gaurav Kumar, Dr. Diwakar Yagyasen, Mr. Surya Vikram Singh. "Voice Assistant Using Python" An International Open Access-revied, Refereed Journal.Unique Paper ID: 152099.

[2] Tulshan, Amrita, Dhage, Sudhir. (2019). “Survey on Virtual Assistant: Google Assistant, Siri, Cortana, Alexa”, 4th International Symposium SIRS 2018, Bangalore, India, September 19–22, 2018, DOI: 10.1007/978-981-13-5758-9.

[3] Abhay Dekate, Chaitanya Kulkarni, Rohan Killedar. (2016). "Study of Voice Controlled Personal Assistant Device", International Journal of Computer Trends and Technology (IJCTT) - Volume 42 Number 2, DOI: 10.14445/22312803/IJCTT-V42P107.

[4] S. Subhash, P. N. Srivatsa, S. Siddesh, A. Ullas and B. Santhosh, "Artificial Intelligence-based Voice Assistant," 2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4), London, UK, 2020, pp. 593-596, DOI: 10.1109/WorldS450073.2020.9210344.

[5] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. arXiv preprint arXiv:2203.02155, 2022.

[6] picovoice.ai. Porcupine Wake Word | Python API. [online]. Available: https://picovoice.ai/docs/api/porcupine-python.

[7] platform.openai.com. OpenAI API Reference. [online]. Available: https://platform.openai.com/docs/api-reference.

[8] pypi.org. SpeechRecognition. [online]. Available: https://pypi.org/project/SpeechRecognition.

[9] pyttsx3.readthedocs.io. pyttsx3-Text-to-speech x-platform. [online]. Available: https://pyttsx3.readthedocs.io/en/latest/index.html.

[10] https://pypi.org/project/py-avataaars/ py-avataaars. [online]. Available: https://pypi.org/project/py-avataaars.

[11] Li Zhou, Daqing Zhang, and Zhongqi Zhang. "Conversational Agents for Task Automation: A Survey", ACM Computing Surveys 2021.

[12] C. Lu, B. Cao, L. Li, P. Padala, and N. D. Lane. "Alexa: A Voice-activated Virtual Assistant", arXiv 2018.

[13] Tara Sainath, Ron Weiss, Kanishka Rao, Bo Li, Pedro Moreno, Eugene Weinstein, and others. "Hey Siri: An On-device DNN-powered Voice Trigger for Apple’s Personal Assistant", Interspeech 2017.

[14] M. I. F. Chowdhury, M. A. Hossain, and A. Muhammad. "Survey of Machine Learning Approaches for Natural Language Processing Tasks in Chatbots", Journal of King Saud University - Computer and Information Sciences 2020.

[15] Sai Prasanna, Jeff Wu, Saurabh Kumar, et al. "DialogGPT: Large-Scale Generative Pre-training for Conversational Response Generation", arXiv 2021.

[16] G. Riccardi, T. Hori, S. Watanabe, et al. "Speech Recognition for Voice Assistants: A Benchmark", arXiv 2021.

[17] Marc Lanctot, Jacob Devlin, and Jelena Luketina. "Towards Lifelong Conversational AI", arXiv 2021.

[18] Xinya Du, Wen-Wei Hsieh, Siqi Sun, et al. "Conversational AI: Building System-Level Intelligence", arXiv 2021.

[19] Andrew L. Maas, Zornitsa Kozareva, Cody Lester, et al. "Zero-Shot Learning in Modern NLP", arXiv 2019.

[20] Shirisha Reddy, Cheela Shreejit, Durvasula Haritha Bhaskara Vignya. (2023). "Emotion Recognition through Text, Speech and Image. International Journal for Research in Applied Science and Engineering Technology". 11. 1493-1506. 10.22214/ijraset.2023.57652.