IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Neural Intelligent Virtual Assistant

[1]Swayam Mane, [2]Kalpesh Kadam, [3]Mahendra Lohar, [4]Minal Lopes
[1,2,3]Student, [4]Assistant Professor
[1]Department of Information Technology,
[1]St. Francis Institute of Technology Borivali, Mumbai Maharashtra

Abstract: In an era marked by the rapid advancement of technology, traditional computer systems with limited functionalities have evolved into highly sophisticated platforms capable of executing diverse tasks, thanks to innovations such as machine learning, artificial intelligence, and deep learning. However, despite these advancements, if users still encounter challenges in interfacing with computers using conventional input devices, the utility of such progress is called into question. To address this gap, we have developed a voice assistant application using Python. This application boasts a range of functionalities, including the ability to initiate commands such as launching software applications like Word and Excel, web application, sending mail , WhatsApp messages and offering real-time information such as current dates, times, and CPU performance metrics.

Index Terms - Voice Assistant, Speech Recognition, Neural Network, Human-Computer Interaction, Voice Recognition.

I. INTRODUCTION

A personalized intelligent voice assistant is a sophisticated software application that utilizes cutting-edge speech recognition and natural language processing (NLP) technologies. By leveraging NLP, these assistants can decipher and interpret the complex structure and intent behind user commands, enabling them to respond appropriately to a wide range of requests. Concurrently, speech recognition technology enables the conversion of spoken language into text, which serves as the input for the system's processing.
These assistants are adept at executing voice-handled tasks such as initiating commands to take screenshots or launch applications, all without requiring manual input from the user. This hands-free interaction capability is made possible by the seamless integration of NLP and speech recognition technologies, which allow users to interact with computer systems or applications effortlessly.
In essence, personalized intelligent voice assistants act as intuitive intermediaries between users and technology, bridging the communicative gap between human language and machine comprehension. By deciphering natural language commands and executing corresponding actions, they provide users with a user-friendly and accessible means of engaging with computer systems and applications. This functionality proves particularly advantageous in contexts where hands-free operation is imperative, such as during driving or multitasking scenarios. Ultimately, voice assistants serve a pivotal role in augmenting user experiences and optimizing workflow efficiency.

- **Information and Entertainment:**

Users can ask voice assistants for news updates, and weather forecasts, and play music, audio books, or podcasts, making them a one-stop source for information and entertainment.

- **Accessibility:**

Voice assistants, such as Siri, Alexa, and Google Assistant, play a crucial role in enhancing accessibility for individuals with disabilities. These tools empower users by providing them with the ability to control their devices and access information using just their voice, eliminating the need for traditional input methods like typing or touching screens. For people with mobility impairments, voice commands offer a hands-free alternative, enabling them to interact with technology independently.

II. LITERATURE REVIEW

Dr. Diwakar Yagyasen, Nivedita Singh et al, explained about Voice Assistant using Python based on the use of several libraries such as NLTK, Pyttsx3, which provide the framework for speech recognition and natural language processing This system, although similar to ours limits its features. System is based on natural feature tracking, and This the model provides limited primary features and functions- abilities, including a GUI. Model makes use of simple APIs such as Pyttsx3, Date Time, etc. Which requires less training resources the model makes use of frameworks such as PyCharm and PyQtPy to create a decent voice interface for its responses[1].

Ayush Chinchane, Aryan Bhushan et al, explained regarding SARA: A Voice Assistant using Python model which uses a more complex approach called the Kaldi speech recognition which consists of a structure of logical loops such as if and while. This model is characterized by a low error rate in speech recognition by making use of complex algorithms such as MLP(Multilayer Perceptron) and IFTTT(If- this-then-that). This model works mainly on two prime technologies, via Wolfram Alpha and Selenium for automated factual query responses. Provision of direct answers to factual queries and integration of browser automation tools and libraries provides the model with a smooth workflow[2].

Deepika patel and Toran Verma had explained Application of Voice Assistant using ML with a combination of LSTM and HMM architectures which provide a base for hidden pattern tracing and deep extraction of meaning in the sentence. The system achieves accuracy for the cost of time latency. This model is trained over more stable algorithms like XgBoost, OCR(Optical Character Recognition) which helps achieve better precision in the responses than the other models. There is a significant increase in the precision of responses but the model suffers from time latency. This model owes its high precision to major technologies CTC-based LSTM acoustic model and a variation of HMM API[3].

Polyakov E.V, Mazhanov M.S, Rolich A.Y. et al, ALICE – Intelligent Voice Assistant for IOT is built on the concept of pattern recognition and voice activation which can be used for authorization purposes This model uses some prime speech recognition and natural language processing algorithms such as ASR, KNN and Bayesian Classifiers. The model is developed focusing on IOT applications and hence has high device connectivity and operability but suffers from reduced accuracy in language interpretation and low precision on positive responses. Voice Biometrics is the major technology used in this model which emphasizes on Voice Activation and Dialog Management[4].

III. PROPOSED SOLUTION

N.I.V.A. (Desktop-based Neural Intelligent Virtual Assistant) distinguishes itself from conventional voice assistants by being tailored specifically for desktop environments. Developed within the Visual Studio Code integrated development environment (IDE), all Python files constituting the project were authored and managed within this environment, ensuring seamless compatibility and ease of development. The project relies on a curated selection of modules and libraries, including but not limited to pyttsx3 for text-to-speech synthesis, Speech Recognition for speech recognition capabilities, Date time for time-related functionalities, Wikipedia for accessing Wikipedia articles, Smtplib for email sending functionality, torch for machine learning tasks, and PyJokes for humorous responses.

The performance evaluation of N.I.V.A. is conducted based on key metrics such as accuracy in understanding and executing user commands, as well as response time, which measures the latency between receiving a command and generating an appropriate response. The architecture diagram of N.I.V.A. , a voice assistant depicts a process flow that starts with speech recognition, followed by natural language

processing (NLP), and comparison of conversation patterns. Based on the outcome of this comparison, N.I.V.A. can generate audio or text outputs, or perform various tasks such as opening applications, taking screenshots, etc. The diagram highlights the key components involved in the voice assistant, including libraries and modules such as Task, speak, listen, random, json, torch, Neural Network.

The algorithm of N.I.V.A. to import necessary libraries and modules such as Task, speak, listen, random, json, torch, and Neural network. Task is a module containing functions that can execute various tasks based on user input. The Speak module contains functions that can generate audio outputs from text. This is useful for providing voice responses to user input. The Listen module contains functions that can listen to user input and convert it to text. This is the first step in the speech recognition process. The Random module provides functions for generating random numbers, which can be useful for generating random responses to user input. Torch is a popular machine learning library that provides tools for building and training neural networks. Neural network module contains functions for tokenizing user input and feeding it into a pre-trained neural network for intent classification. These libraries and modules are used for various purposes such as speech recognition, natural language processing, executing tasks, generating audio outputs, and more. To load the intents .json file that contains various intent tags, patterns, and responses. This file serves as a reference for the system to understand the user's input and execute the corresponding task. It contains patterns that the system looks for in user input to match with the intent tags and responses that the system generates based on the matched intent tag. To load the pre-trained neural network model that was trained on a dataset containing examples of user input and the corresponding intent tags. In order to accurately understand and respond to user input, N.I.V.A. uses a pre-trained neural network model.

N.I.V.A. is a type of artificial neural network that has been trained on a dataset of user inputs and their corresponding intent tags. During the training process, the neural network learns to recognize patterns and relationships between words and intent tags. When a user speaks to the voice assistant, their input is processed and passed through the neural network model. The model analyzes the input and predicts the corresponding intent tag. The pre-trained neural network model is important because it allows N.I.V.A. to accurately understand the user's intent and respond appropriately. Without the model, the system would have to rely on simple keyword matching or rules-based approaches, which can be less accurate and more prone to errors. By using a neural network model, N.I.V.A. can make more accurate predictions based on that data. To define a function named Main that listens to user input, classifies it into an intent tag using the pre-trained neural network, and executes the corresponding task based on the intent tag. This function forms the core of the voice assistant and uses various modules and functions to execute the user's request. To get user input by calling the listen function from the Listen module. This module uses speech recognition techniques to convert the user's spoken words into text. To tokenize the user input using the tokenize function from the Neural network module.

Tokenization is a fundamental step in natural language processing that involves breaking down a piece of text into smaller units, typically words, called tokens. It is a crucial step in NLP because many NLP algorithms, including machine learning models, require input data to be in the form of a sequence of tokens rather than as plain text. In the context of the N.I.V.A., the tokenize function from the Neural network module is used to convert the user's spoken input into a list of individual words or tokens. The function uses regular expressions to split the input text into individual words, removing any punctuation or unnecessary characters, and returns a list of tokens. To pass the tokens to the pre-trained neural network to match the intent tag from the tags list. The network predicts the intent tag based on the input tokens. To check for specific keywords in the response, such as "time", "date","screenshot","CPU","weather","jokes","wikipedia","google","open","close","news","switchwindow", "searchfile","create","record","summarization".

N.I.V.A. system checks for specific keywords in the user's input to determine the appropriate task to execute. These keywords are predefined and are used as triggers to perform specific actions. For example, if the user says "What is the weather like today?", the N.I.V.A. system recognizes the keyword "weather" and executes the task of retrieving the weather information for the current day. Similarly, if the user says "Open Google Chrome", the system recognizes the keyword "open" and opens the Google Chrome application. These keywords help identify the user's intent and execute the appropriate task. By recognizing these keywords, the system can respond quickly and accurately to the user's requests, making the voice assistant more useful and efficient.

To execute the corresponding task using the InputExecution or NonInputExecution function from the Task module. This module contains various functions that perform specific tasks such as opening applications, taking screenshots, retrieving weather information, and more. To use the Say function from the speak module to speak a random response from the intent tag's responses list. This module generates an audio output that provides the user with the information they requested.

IV. RESULTS AND DISCUSSION

4.1 Results

The below metrics are assessed based on the system's ability to accurately interpret and respond to user commands across various tones and accents.

Table 4.1: Performance Metrics

Accuracy	71%
Response Time	3 Seconds Maximum
Error Rate	27%
Task Completion Rate	80%

4.2 Implementation

4.2.1 Whatsapp Messaging

The system will prompt the user to input the recipient's name along with the message they want to send through a voice command interface. Subsequently, it will provide information about the scheduled delivery time of the message to the recipient. The system will automate the process of sending a message to the recipient at a scheduled time. It will trigger the opening of the WhatsApp application window automatically at the specified delivery time. Additionally, the scheduled delivery time will be displayed within the WhatsApp interface for the user to see. Furthermore, within the WhatsApp interface, the scheduled delivery time prominently displays, empowering users with awareness of the impending message transmission. This feature enhances user experience by offering transparency and control over message delivery timing.

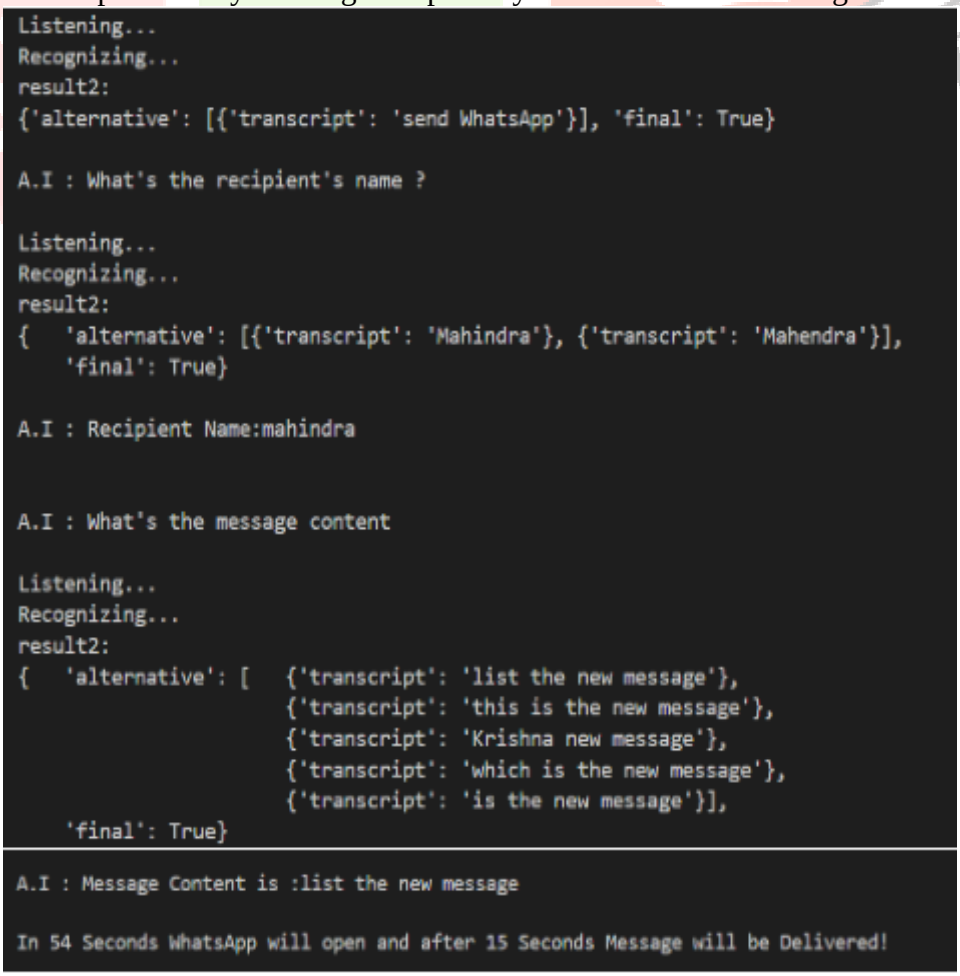

Fig 4.2.1 Whatsapp Messaging

4.2.2 Email

Upon receiving a voice command from the user to send an email, the N.I.V.A. system will assist by initiating the email delivery process. It will prompt the user to provide the subject for the email interaction. Following the subject input, N.I.V.A. will prompt the user to provide the email body and recipient's address. Upon receiving the recipient's email address, N.I.V.A. will verify it by requesting user confirmation. This step ensures accurate delivery and enhances user experience. Once verified, the system proceeds to compose the email with the provided content and sends it to the intended recipient. Upon successful delivery, N.I.V.A. confirms the completion of the email sending process to the user.

```
Listening...
Recognizing...
result2:
{'alternative': [{'transcript': 'send email'}], 'final': True}
you :send email.
Listening...
Recognizing...
result2:
{   'alternative': [   {'transcript': 'send email'},
                       {'transcript': 'send image'},
                       {'transcript': 'send emails'}],
    'final': True}

A.I : Hello, How can I assist you with your email?


A.I : What is the subject of the email?

Listening...
Recognizing...
result2:
{   'alternative': [   {'transcript': 'Niva AI'},
                       {'transcript': 'Nibha AI'},
                       {'transcript': 'neeva AI'},
                       {'transcript': 'Niva a i'},
                       {'transcript': 'Nibha a i'}],
    'final': True}
```

```
A.I : What should be the body of the email?

Listening...
Recognizing...
result2:
{   'alternative': [   {'transcript': 'and virtual assistant'},
                       {'transcript': 'in virtual assistant'},
                       {'transcript': 'when virtual assistant'}],
    'final': True}

A.I : To whom should I send the email?

Listening...
Recognizing...
result2:
{   'alternative': [{'transcript': 'MN Lohar'}, {'transcript': 'MN Loha'}],
    'final': True}

A.I : mnlohar@gmail.com


A.I : Is this the correct email address? Say send it  or no.
```

```
Listening...
Recognizing...
result2:
{'alternative': [{'transcript': 'send it'}], 'final': True}

A.I : send it


A.I : Email sent successfully.
```

Fig 4.2.2 Email

4.2.3 Summarization

N.I.V.A. is designed to prompt users to input the text content they wish to summarize. This content can be in the form of a paragraph, which N.I.V.A. will then condense into a concise summary comprising 5 to 7 lines. Additionally, N.I.V.A. will provide this summarized paragraph in voice format as a response to the user's input.

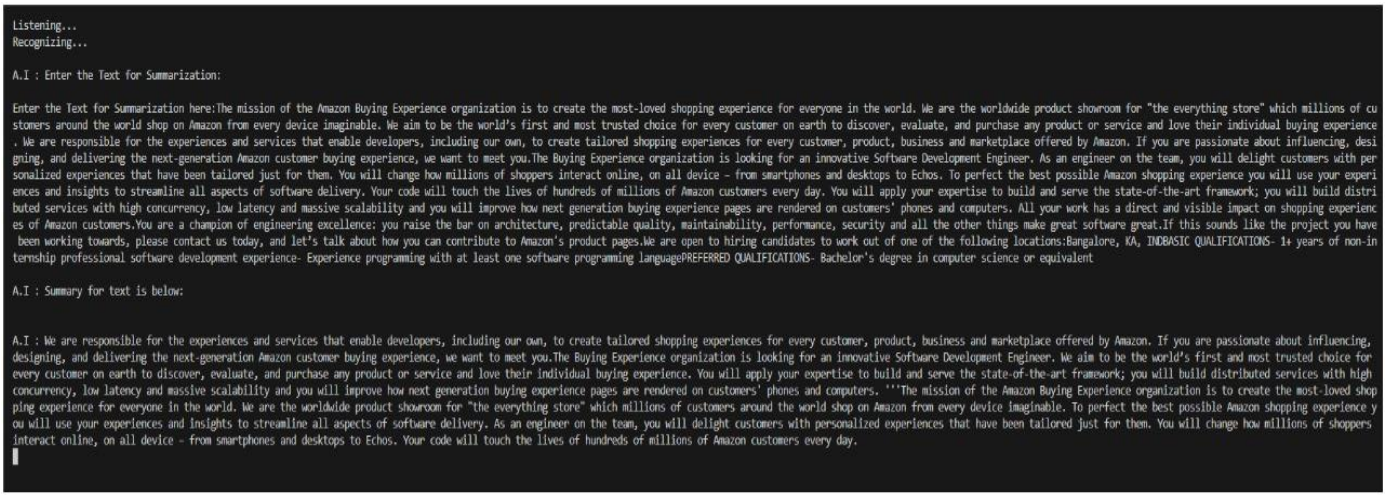

Fig 4.2.3 Summarization

4.2.4 Switching Window

When a user commands N.I.V.A. to switch the current window, the system engages in voice recognition to interpret the user's spoken instruction. Then executes the command by manipulating the graphical user interface to transition focus from the current application window to another specified window. Technically, this process involves interfacing with the operating system's window management functions to identify and activate the designated window based on the user's input.

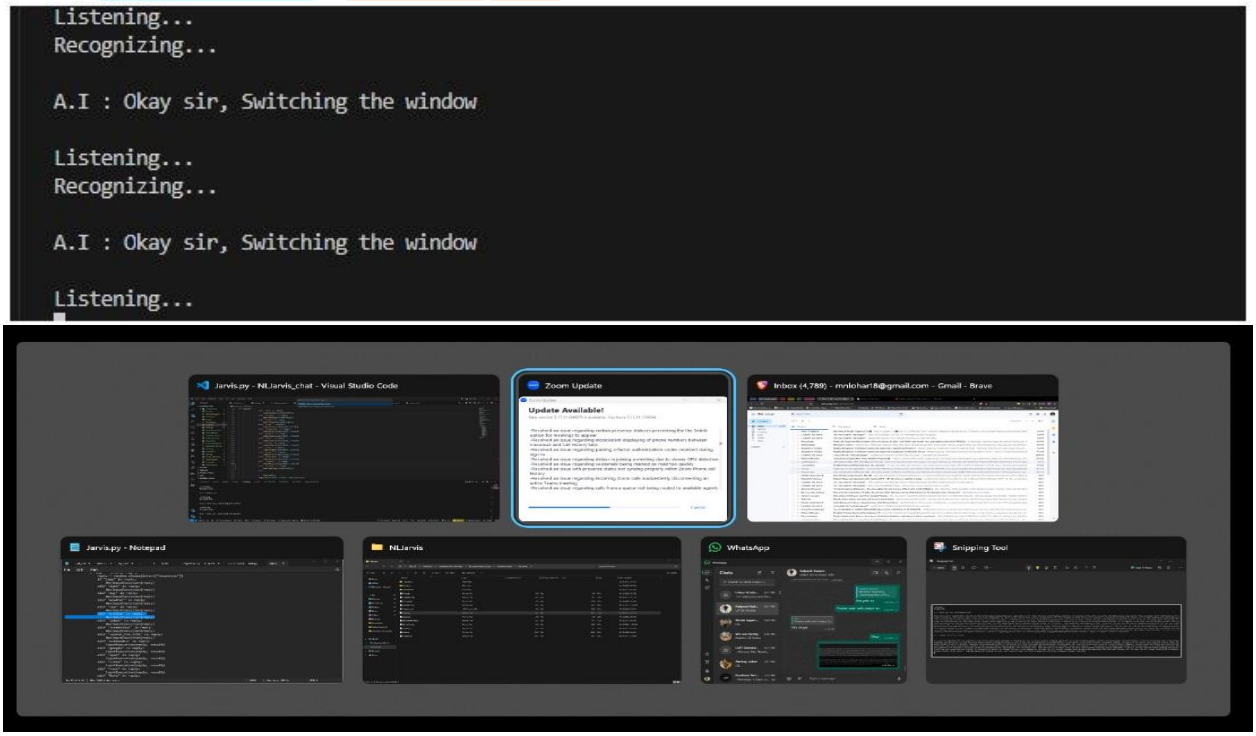

Fig 4.2.4 Switching Window

4.2.5 Chat History

N.I.V.A. retains a log of all recent interactions, including chat messages and voice commands, in its back-end database. Users have the capability to access this stored history, allowing them to review previous interactions and actions initiated through voice commands. Technically, this involves maintaining a structured database that records each interaction in chronological order, associating them with user sessions or unique identifiers. The back-end system facilitates efficient retrieval of this historical data, enabling users to track and revisit their past interactions with N.I.V.A., including any commands issued via voice input.

```
2024-02-25 05:59:54 - Assistant: Here's your Assistant sir
2024-02-25 06:00:57 - User: hello jarvis
2024-02-25 06:00:57 - Assistant: How are you2024-03-10 20:24:15 - User: hello jarvis
2024-03-10 20:24:15 - Assistant: How are you sir
2024-03-10 20:28:04 - User: hello jarvis
2024-03-10 20:28:04 - Assistant: Here's your Assistant sir
2024-03-10 20:30:29 - User: hello jarvis
2024-03-10 20:30:29 - Assistant: Hello sir
2024-03-10 20:35:10 - User: hello jarvis
2024-03-10 20:35:10 - Assistant: How are you sir
2024-03-10 20:44:49 - User: hello
2024-03-10 20:44:49 - Assistant: Hello
2024-03-10 20:44:56 - User: switch a window
2024-03-10 20:44:56 - Assistant: window
2024-03-10 20:45:09 - User: switch window for me
2024-03-10 20:45:09 - Assistant: window
2024-03-10 21:20:21 - User: hello jarvis
2024-03-10 21:20:21 - Assistant: Hello sir
2024-03-10 21:20:31 - User: switch a window from
2024-03-10 21:20:31 - Assistant: window
2024-03-10 21:21:02 - User: switch off window for me
2024-03-10 21:21:02 - Assistant: window
```

Fig 4.2.5 Chat History

4.2.6 Search File

Upon issuing a voice command to N.I.V.A. (Neural Interactive Virtual Assistant), the system engages the user in a dialogue to specify the name of the file to be searched. Following the user's vocal input, N.I.V.A. proceeds to execute a search operation within the local storage of the user's personal computing environment, scouring through directories and files to pinpoint the exact location of the specified file.

```
Listening...
Recognizing...

A.I : Tell the file name

Listening...
Recognizing...
you :resume.
Found in C:/:
C:/Users\Krishna\AppData\Local\BraveSoftware\Brave-Browser\User Data\Default\IndexedDB\https_resume.naukri.com_0.indexeddb.leveldb
C:/Users\Krishna\AppData\Local\BraveSoftware\Brave-Browser\User Data\Default\IndexedDB\https_resume.naukri.com_0.indexeddb.leveldb\resume
C:/Windows\WinSxS\amd64_microsoft-windows-b..os-resume.resources_31bf3856ad364e35_10.0.22621.1_en-us_ac04201b78086e3e
C:/Windows\WinSxS\amd64_microsoft-windows-b..vironment-os-resume_31bf3856ad364e35_10.0.22621.3085_none_4d8a6ee59bfd20f8
C:/Windows\WinSxS\amd64_microsoft-windows-b..vironment-os-resume_31bf3856ad364e35_10.0.22621.3235_none_4d6fcd9b9c11da27
C:/Windows\WinSxS\amd64_microsoft-windows-b..os-resume.resources_31bf3856ad364e35_10.0.22621.1_en-us_ac04201b78086e3e\resume
C:/Windows\WinSxS\amd64_microsoft-windows-b..vironment-os-resume_31bf3856ad364e35_10.0.22621.3085_none_4d8a6ee59bfd20f8\resume
C:/Windows\WinSxS\amd64_microsoft-windows-b..vironment-os-resume_31bf3856ad364e35_10.0.22621.3085_none_4d8a6ee59bfd20f8\f\resume
C:/Windows\WinSxS\amd64_microsoft-windows-b..vironment-os-resume_31bf3856ad364e35_10.0.22621.3085_none_4d8a6ee59bfd20f8\r\resume
C:/Windows\WinSxS\amd64_microsoft-windows-b..vironment-os-resume_31bf3856ad364e35_10.0.22621.3235_none_4d6fcd9b9c11da27\resume
C:/Windows\WinSxS\amd64_microsoft-windows-b..vironment-os-resume_31bf3856ad364e35_10.0.22621.3235_none_4d6fcd9b9c11da27\f\resume
C:/Windows\WinSxS\amd64_microsoft-windows-b..vironment-os-resume_31bf3856ad364e35_10.0.22621.3235_none_4d6fcd9b9c11da27\r\resume
C:/Windows\WinSxS\Temp\InFlight\0f9ca1e4d26bda0194030000b48f7093\amd64_microsoft-windows-b..vironment-os-resume_31bf3856ad364e35_10.0.22621.3007_none_4d823f059c048982
C:/Windows\WinSxS\Temp\InFlight\0f9ca1e4d26bda0194030000b48f7093\amd64_microsoft-windows-b..vironment-os-resume_31bf3856ad364e35_10.0.22621.3007_none_4d823f059c048982\resume
C:/Windows\WinSxS\Temp\InFlight\0f9ca1e4d26bda0194030000b48f7093\amd64_microsoft-windows-b..vironment-os-resume_31bf3856ad364e35_10.0.22621.3007_none_4d823f059c048982\f\resume
C:/Windows\WinSxS\Temp\InFlight\0f9ca1e4d26bda0194030000b48f7093\amd64_microsoft-windows-b..vironment-os-resume_31bf3856ad364e35_10.0.22621.3007_none_4d823f059c048982\r\resume
C:/Windows\WinSxS\Temp\InFlight\b386eaf6445fda016403000586f8c19\amd64_microsoft-windows-b..vironment-os-resume_31bf3856ad364e35_10.0.22621.2792_none_4d40022d9c34b3dc
C:/Windows\WinSxS\Temp\InFlight\b386eaf6445fda016403000586f8c19\amd64_microsoft-windows-b..vironment-os-resume_31bf3856ad364e35_10.0.22621.2792_none_4d40022d9c34b3dc\resume
C:/Windows\WinSxS\Temp\InFlight\b386eaf6445fda016403000586f8c19\amd64_microsoft-windows-b..vironment-os-resume_31bf3856ad364e35_10.0.22621.2792_none_4d40022d9c34b3dc\f\resume
C:/Windows\WinSxS\Temp\InFlight\b386eaf6445fda016403000586f8c19\amd64_microsoft-windows-b..vironment-os-resume_31bf3856ad364e35_10.0.22621.2792_none_4d40022d9c34b3dc\r\resume
resume not found in D:/
```

Fig 4.2.6 Search File

4.2.7 Create Document

Upon vocalizing a command to N.I.V.A., the system triggers the launch of Microsoft Word application from the local system. Once initiated, N.I.V.A. awaits further instructions for document creation. Users can then dictate the desired content to be written into the document, leveraging the functionality provided by Microsoft Word for text composition and editing. Upon successfully initializing the Microsoft Word document through N.I.V.A., the system enters a state where it awaits user input regarding the content to be included in the document. Users can verbally communicate the desired message or content they wish to write, which N.I.V.A. will then transcribe into text within the open Word document using voice recognition.

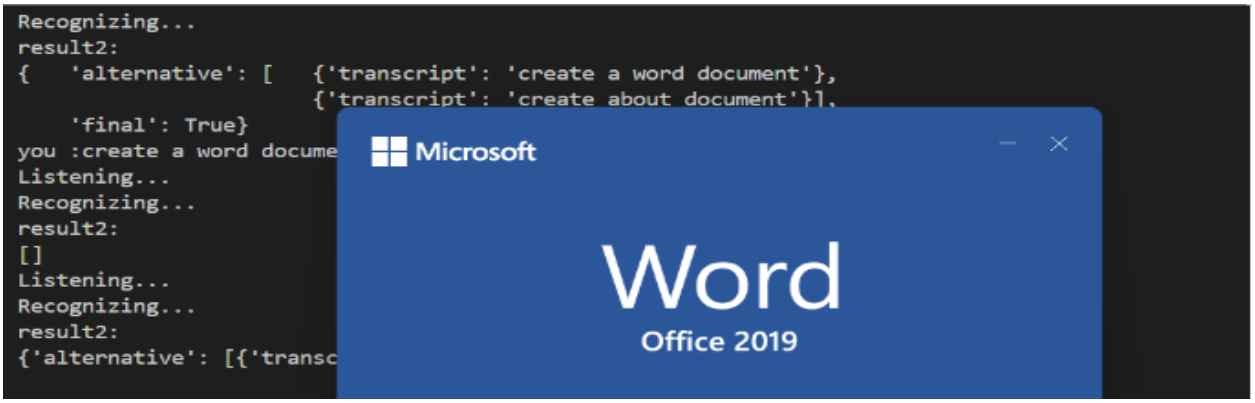

Fig 4.2.7 Create Document

V. CONCLUSION

The rapid evolution of technology, propelled by advancements in machine learning, artificial intelligence, and deep learning, has transformed traditional computer systems into highly sophisticated platforms capable of executing diverse tasks. However, the persistence of challenges in user-computer interfacing with conventional input devices raises questions about the practical utility of such progress. To bridge this gap, our development of a Python-based voice assistant application offers a solution from launching software applications to sending emails and WhatsApp messages, as well as providing real-time information such as dates, times, and CPU performance metrics, this voice assistant represents a tangible response to the need for more intuitive and efficient user-computer interactions in the modern technological landscape.

REFERENCES

[1] Pooja C. Goutam, Monika S. Jalpure, Akshata S. Gavade, Pranjali Chaudhary, Prof. A.V. Gundavade, "Voice Assistant using Python", International Journal of Creative Research Thoughts(IJCRT), Volume 10 Issue 6, June 2022.

[2] Ayush Chinchane, Aryan Bhushan, Ayush Helonde, Prof. Kiran Bidua, "SARA: A Voice Assistant Using Python", International Journal for Research in Applied Science & Engineering Technology (IJRASET), Volume 10 Issue 6, June 2022

[3] Deepika Patel, Toran Verma, "Application of Voice Assistant using Machine Learning: A Comprehensive Review", Advances and Applications in Mathematical Sciences,Volume 21 Issue 9, July 2022

[4] Polyakov E.V., Mazhanov M.S., Rolich A.Y., Voskov L.S., Kachalova M.V., Polyakov S.V, "ALICE: Investigation and Development of the Intelligent Voice Assistant for the Internet of Things Using Machine Learning", Moscow Workshop on Electronic and Networking Technologies (MWENT), 2018

[5] Rajat Sharma, Adweteeya Dwivedi, "JARVIS - AI Voice Assistant" , International Journal of Science and Research (IJSR), Volume 11 Issue 5, May 2022

[6] Dhanraj K.V.,Kriplani L. and Mahajan S., 'Research Paper on Desktop Voice Assistant', International Journal of Research in Engineering and Science (IJRES),Vol 10 ,Issue 2, July 2021

[7] E. D. Jarvis, "Erich D. Jarvis,"Current Biology, vol. 30, no. 5 , pp. R202–R203, Mar. 2020

[8] S. S. Chowdhury, A. Talukdar, A. Mahmud and T. Rahman, "Domain Specific Intelligent Personal Assistant with Bilingual Voice Command Processing," TENCON 2018 - 2018 IEEE Region 10 Conference, Jeju, Korea (South), 2018

[9] T. Kim, "Short Research on Voice Control System Based on Artificial Intelligence Assistant," International Conference on Electronics, Information, and Communication (ICEIC), Barcelona, Spain, 2020

[10] K. N., R. V., S. S. S., and D. R., "Intelligent Personal Assistant - Implementing Voice Commands enabling Speech Recognition," International Conference on System, Computation, Automation, and Networking (ICSCAN), Pondicherry, India, 2020

[11] S. Kumari, Z. Naikwadi, A. Akole, and P. Darshankar, "Enhancing College Chat Bot Assistant with the Help of Richer Human- Computer Interaction and Speech Recognition,"International Conference on Electronics and Sustainable Communication Systems (ICESC), Coimbatore, India,2020

[12] R. Sangpal, T. Gawand, S. Vaykar, and N. Madhavi, "JARVIS: An interpretation of AIML with the integration of gTTS and Python," 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies(ICICICT), Kannur, India,pp 20