ISSN No:-2456-2165

AI Voice Assistant using NLP and Python Libraries (M.A.R.F)

Roshini Gudla; Petla Mohit Kumar; Farheen shaik; Anish Varin Pilla
GITAM UNIERSITY, Vizag, India

Abstract:- "M.A.R.F" an advanced AI voice assistant chatbot aimed at increasing productivity for professionals, students, and others dealing with modern time management issues. The study article illustrates Jarvis' solution-oriented design in the setting of rapid technology innovation. It addresses time management complexities strategically through task automation, targeted answers, and seamless integration with a variety of technological platforms. Specific functionality such as Google and YouTube search, music playback, chat capabilities, and information retrieval are listed in the abstract. Its potential societal impact is envisioned as a cornerstone solution for increasing productivity, catering to different user segments, and alleviating time-related stress. Jarvis' user-centric benefits are highlighted, emphasizing seamless integration and task automation, promoting it as a bridge between consumers and sophisticated technology. The abstract stresses Jarvis' methodological rigor by proposing accuracy assessment metrics like as precision, recall, and F1-score, which focus thorough evaluation of its prediction accuracy and adaptability. The abstract effectively conforms with scientific standards, clearly communicating objectives, methodology, and predicted effects while displaying scholarly rigor in a way that resonates with the scientific reviewer. The project was built in Python using the pptx3 and pywhatkitlibraries.

Keywords:- *Natural Language Generation, Conversational AI, Natural Language Generation.*

I. INTRODUCTION

Artificial Intelligence has ushered in a new era of human-computer interaction. Chat bots, in particular, have gained prominence for their ability to tackle a diverse range of tasks. Jarvis, our AI chat bot, stands as a testament to this advancement, showcasing a spectrum of capabilities, from music playback and application launching to weather forecasts and audiobook reading.

A. Voice Processing

Jarvis leverages advanced voice recognition technology, harnessing the Speech Recognition library to interpret spoken commands. This enables seamless natural language interaction. The pyttsx3 library facilitates lifelike voice synthesis, enhancing the conversational experience.

B. Web Interaction and Data Retrieval

Web browsing and data retrieval are integral to Jarvis's functionality. It harnesses Python libraries such as web browser, BeautifulSoup, and pywhatkit to navigate the web, extract information from websites, and provide answers to user queries. Jarvis can scour Wikipedia, retrieve general knowledge insights, and scrape data for how-to guides.

C. Multilingual Proficiency

Jarvis boasts multilingual proficiency, courtesy of the Google Translate API. It can engage in conversations and provide content in various languages. Additionally, Jarvis can read books in different languages and offer translations upon request.

D. Automation and System Control

Task automation forms a core aspect of Jarvis's utility. Itcan initiate actions like application management, YouTube control, screen capture, and system shutdown using keyboard shortcuts and system commands.

E. Data Storage and Retrieval

Jarvis incorporates a memory feature, permitting users to store and retrieve information effortlessly. It can remember notes, set reminders, and adapt to user preferences for futureinteractions.

F. Entertainment and Recreation

In addition to its practical applications, Jarvis brings an element of entertainment by sharing jokes and facilitating music playback from local libraries or YouTube searches.

G. Customization and Expansion

Jarvis is highly adaptable, designed to accommodate user-specific requirements. It can be extended with new functionalities to cater to diverse needs.

In conclusion, Jarvis, our Python-based AI chat bot, embodies a comprehensive suite of features and capabilities. It serves as a versatile virtual assistant, adept at addressing inquiries, offering information, providing entertainment, and automating tasks. Harnessing the potential of AI, Jarvis is poised to enhance digital interactions, making them more engaging and efficient.

II. LITERATURE SURVEY

Smith, J et al.,[8] focuses on developing a chatbot using deep neural learning for potential applications, especially in the medical field. It uses Kaggle data, experiments with optimization algorithms and weight initialization, emphasizing their impact on accuracy. A user-friendly GUI is created. While chatbots can enhance interactions, challenges like understanding emotions and privacy concerns remain.

Johnson, A. et al.,[2] creates an intelligent voice recognition chatbot using a web service framework. It uses a black box approach, ensuring smooth client-server communication through a user-friendly interface for text and voice input. The system utilizes Java and AIML for voice processing, integrating AI from ALICE, and employs a third-party expert system for handling complex queries. While its modular, distributed design enables scalability, potential longevity issues arise with the expert system.[2]

P. Kulkarni et al .,[4]does a research study on Conversational AI, focusing on its architecture and components.It reviews advanced machine these components. Study explores potential applications in healthcare, education, tourism, and more. Overall, its goal is to contribute to the understanding of Conversational AI's evolution and provide a foundation for future research and innovation.

M. Kathirvelu et al., [5] implements a chatbot using the BiLSTM model in Python to optimize responses to user inquiries, particularly in customer service. It utilizes the Cornell Movie Dialog Corpus for data preparation and achieves remarkable 99.52% average accuracy through model selection and parameter tuning. This work showcases the practicality of advanced natural language processing and paves the way for future chatbot enhancements.

P. Thosani et al., [9] aims to enhance conversational AI bots to better understand and fulfill user service requirements. It addresses challenges such as incomplete and uncertain user expressions and diverse service resources. The project uses a fine-tuned BERT model for precise user intention recognition. It introduces a granular computing method to efficiently prune user requirements, reducing conversation rounds and improving user experience.

Ultimately, the goal is to make conversational AI bots more adaptable to complex scenarios, enabling more natural and intelligent interactions with users.

III. METHODOLOGY

In the development of a comprehensive voice-controlled assistant, we plan to integrate a wide range of functionalities. First and foremost, we aim to achieve seamless speech recognition and synthesis integration, utilizing libraries like Speech Recognition and pyttsx3 to enable users to interact with the assistant through spoken commands and receive synthetic speech responses. Web interaction capabilities are a crucial part of our assistant, allowing it to open websites, search the web, and extract information from websites, all powered by libraries such as web browser, requests, Beautiful Soup, and pywhatkit.

Additionally, we will implement voice-controlled web browsing for hands-free navigation.

The assistant's versatility extends to application control, enabling it to launch, close, or interact with various software applications installed on the system using the os module.

Information retrieval capabilities will be a key feature, with the assistant fetching data from sources like Wikipedia, Google, and WikiHow. Multilingual support through the Google Translate API will enhance accessibility, allowing users to interact with the assistant in multiple languages.

File handling, media playback, automation, location-based services, data persistence, user interface development, error handling, and security and privacy considerations are all integral components of our assistant's design. Furthermore, we will focus on documentation, scalability, user-centered design, and ethical considerations to ensure a well-roundedand responsible AI assistant.

To summarize, our voice-controlled assistant will encompass a wide array of functionalities, from speech recognition to application control, web interaction, information retrieval, and beyond. It will prioritize user-friendliness, security, privacy, and ethical considerations, offering a robust and accessible tool for users while also accommodating potential future advancements in AI capabilities. Comprehensive documentation and usability testing will ensure a positive user experience, making the assistant a valuable addition tousers' daily lives.

IV. FUNCTIONALITY OF THE CHAT BOT

In the context of developing a chatbot, this code serves as the foundation for implementing a voice-controlled chatbot. Here's how you can map the functionalities in the code to different chatbot development techniques and approaches:

A. Rule-Based Conversation:

In the TaskExe function, you can see that the code listens for specific voice commands (e.g., "hello," "how are you," "music," "wikipedia," etc.) and responds with predefined actions. This is similar to a rule-based chatbot, where specific patterns or commands trigger predefined responses.

B. Deep Learning:

While the code does not explicitly use deep learning for natural language understanding, you could integrate deep learning models like speech recognition (e.g., using deep learning-based ASR models) and natural language understanding (e.g., using neural network-based intent recognition models) to enhance the chatbot's ability to

ISSN No:-2456-2165

understand and respond to natural language.

C. Domain-Specific Chatbot:

You can adapt this code to create a domain-specific chatbot by customizing its responses and actions for a specific domain or use case. For example, you could modify it to provide information about a particular industry or field.

D. Chatbot Builders:

While the provided code is not created using a chatbot builder tool, you could leverage chatbot builder platforms to develop similar voice-controlled chatbots without coding, using drag-and-drop interfaces. These platforms often allow for easy integration of natural language understanding and other AI capabilities.

E. Ensemble Methods:

You can enhance the chatbot's functionality by combining rule-based, retrieval-based, and generative methods. For instance, you can incorporate rule-based responses for specific commands while using deep learning-based models for more complex natural language interactions.

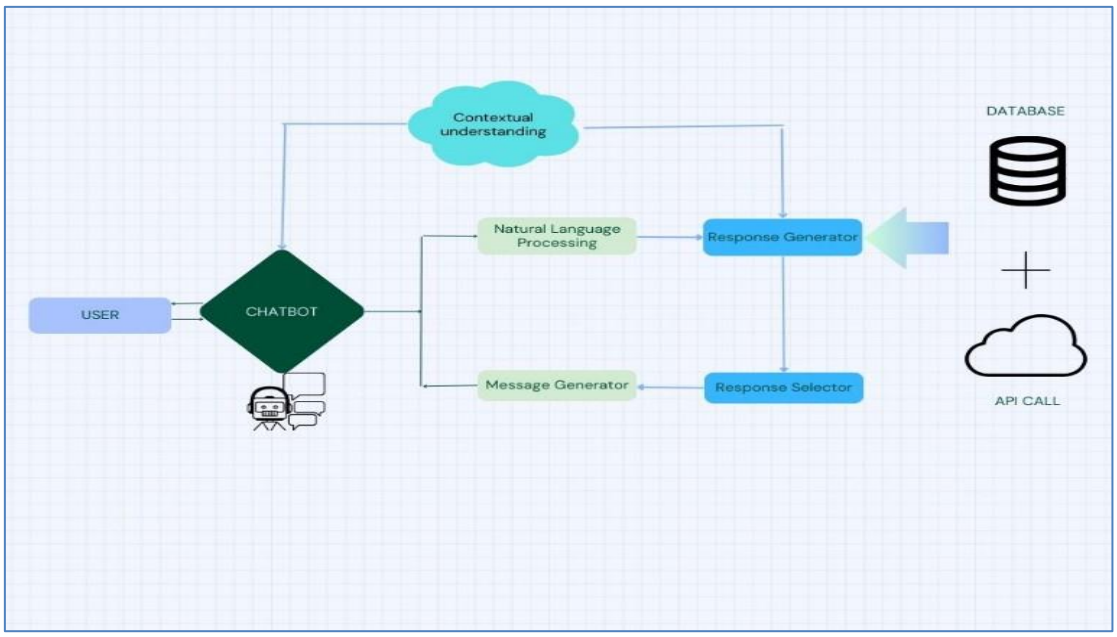

Fig. 1: Flowchart of Chatbot

V. OVERVIEW OF TECHNOLOGY

This project necessitates the utilization of several pivotal technologies, including Python for core programming, speech recognition libraries (e.g., SpeechRecognition), text-to-speech conversion capabilities (via pyttsx3), and potentially cloud-based services such as Google Cloud for advanced speech recognition capabilities.

Within the purview of specialized fields, this project resides Its overarching objective is the creation of an intelligent and user-friendly interface that comprehends and responds to human language and commands.

Technical terminology germane to this project encompasses Speech-to-Text, Text-to-Speech, Natural Language Generation, User Interface Design, Application Development, and Cloud Computing.

This project aspires to fashion an advanced voice-controlled personal assistant by rectifying present constraints and integrating leading-edge technologies from the realms of NLP and HCI, all while prioritizing user customization and the seamless incorporation of user feedback.

- **Speech Recognition and Text-to-Speech: pyttsx3:** Utilized for text-to-speech conversion, enabling the AI assistant to communicate with users through voice.
- **Speech Recognition:** Empowered with speech recognition capabilities, allowing the assistant to understand and process voice commands.

A. Web Interaction:

- **Web browser:** Employed for seamless web browsing, enabling actions such as launching websites and performing web searches.
- **Web Scraping and Data Extraction: Beautiful Soup (bs4):** Used for web scraping, extracting essential data from web pages.
- **requests:** Facilitates making HTTP requests to websites, critical for fetching data like temperature information.

B. Media and Entertainment:

- **pywhatkit:** Enables music playback and YouTube video searches, enhancing user entertainment.
- **playsound:** Used for playing audio files, including alarms.
- **pyjokes:** Fetches jokes to provide users with a touch of humor.

C. Information Retrieval:

- **wikipedia:** Empowered to retrieve information from Wikipedia, enriching the assistant's knowledge base.
- **pywikihow:** Supports searching and retrieving how-to articles for informative responses.
- **Translation and Multilingual Support: googletrans:** Used for language translation, enabling the assistant to comprehend and respondin various languages.

D. File Handling and Automation:

- **OS:** Manages file operations and system commands, including launching applications, closing processes, and handling files.
- **PyPDF2:** Facilitates interaction with PDF files, allowing text extraction and translation. **pytube:** Enables YouTube video downloads, enhancing the assistant's capabilities.
- **Graphical User Interface (GUI):** tkinter: Creates a user-friendly graphical interfacefor specific tasks, such as video downloading.
- **Date and Time Management:** Datetime: Supports date and time-related functions,including setting alarms based on user-defined times.

E. Keyboard Automation:

- **Keyboard:** Simulates keyboard inputs for controlling actions in the YouTube player and automating tasks within the Chrome browser.
- **Natural Language Processing (NLP):** The code implicitly utilizes NLP for interpreting user voice commands and generating appropriate responses.
- **Cloud Services and External APIs:** The code may interact with external cloud-based services, such as Google, for web searches and dataretrieval, enhancing its functionality.

VI. IMPLEMENTATION CODING

Incorporating the M.A.R.F voice assistant into educational environments involves leveraging Python for creating a user-friendly interface and SQL for managing the database interactions. This process is demonstrated through the development of two key functionalities: View Timetable and Edit Timetable. The View Timetable function enables users to inquire about their schedules for any given day using voice commands, effectively parsing these commands to fetch and display the relevant timetable from the database. Conversely, the Edit Timetable function allows users to update their schedules, illustrating the assistant's capabilityto interact with the database based on user input. This integration exemplifies the practical application of AI in automating and streamlining educational processes, showcasing the potential for enhancing efficiency and user engagement through advanced technology solutions.

- str (String):Used to store text and characters. For example, user voice commands, application names, and URLs are stored as strings.
- int (Integer): Used for integer values, primarily for page numbers, time, and numeric input.
- float (Floating-point): Not explicitly used in thecode, but it's a data type for decimal numbers.
- bool (Boolean): Represents either True or False. In the code, it's used in conditional statements.
- list: A collection of items, used to store predefined song names and manage web browser history.
- dict (Dictionary): Not explicitly used, but it's a datatype that stores key-value pairs, useful for mapping values or settings.
- class: Custom classes are used to define and organize related functions, like the TaskExe function that encapsulates various assistant tasks.

There are various data types, string lists, and data used to create a voice-controlled personal assistant program. Here's a list of these elements and a brief explanation of each:

➢ *Data Types:*

- String: Strings are used extensively throughout the code to store and manipulate text data. They are enclosed in single or double quotes and represent user commands, text extracted from PDFs, URLs and more.
- Integer: Integers are used for numerical operations, such as specifying the number of pages in a PDF book and page numbers.
- Float: Floats are used for specifying the rate of speech (e.g., `engine.setProperty('rate', 170)`).
- Boolean: Booleans are not explicitly used in thecode, but they can be used for conditions and decision-making.
- Lists: Lists are used to store predefined song names, which can be played upon user request(e.g., `"akeli"` and `"blanko"` in the `Music()`function).

➢ *String Lists:*

- `voices`: This is a list of available voices for the text-to-speech engine. It is used to select a voice for the assistant.
- `query`: The `query` variable is a string that stores the user's voice command after speechrecognition.
- `music Name`: This string stores the name of thesong to be played in the `Music()` function.
- `name`: This string stores the name of a website to be opened in the `OpenApps()` function.
- `line`: A string containing the line to betranslated in the `Tran()` function.
- `remeberMsg`: A string storing a reminder message provided by the user.
- `op`: A string storing a how-to query used in the `TaskExe()` function.

➢ *Other Data:*

- `engine`: An instance of the `pyttsx3` text-to-speech engine used for converting text to speech.
- `r`: An instance of the `Recognizer` class from the `speech_recognition` library for capturing audio.
- `web browser`: A module used for opening web pages.
- `Beautiful Soup`: A class for parsing HTML content.
- `pywhatkit`: A library for performing various tasks, including playing YouTube videos.
- `wikipedia`: A library for querying Wikipedia articles.
- `Translator`: An instance of the `Translator` class from

the `googletrans` library for language translation.
- `os`: A module for interacting with the operating system, used for starting files and applications.
- `pyautogui`: A library for taking screenshots.
- `psutil`: A library for retrieving system information.
- `Tk`: A class from the `tkinter` library for creating a graphical user interface.
- `StringVar`: A class for holding and manipulating string variables in `tkinter`.
- `PyPDF2`: A library for working with PDF files.
- `YouTube`: A class from the `pytube` library for downloading YouTube videos.
- `datetime`: A module for working with date and time.
- `playsound`: A library for playing audio files.
- `keyboard`: A library for controlling keyboard input.
- `pyjokes`: A library for generating jokes.

These data types and string lists are essential components of the code, enabling the voice-controlled assistant to understand and execute user commands while interacting with various libraries and modules to perform a wide range of tasks.

VII. TESTING

Here are some test case scenarios you can use to evaluate MARF. These test cases cover a range of functionalities voice assistant:

A. Test Case 1: Greeting
- Input: "Hello, Jarvis."
- Expected Output: "Hello Sir, I am Jarvis, your personal AIassistant. How may I help you?"

B. Test Case 2: Basic Commands
- Input: "What's the weather like today?"
- Expected Output: Provides the current weather for the default location (e.g., Delhi).

C. Test Case 3: Music Playback
- Input: "Play the song 'Akeli'."
- Expected Output: Starts playing the song "Akeli."

D. Test Case 4: Application Launch
- Input: "Open Microsoft VS Code."
- Expected Output: Launches Microsoft VS Code.

E. Test Case 5: Web Browsing
- Input: "Open YouTube."
- Expected Output: Opens the YouTube website in the default web browser.

F. Test Case 6: Wikipedia Search
- Input: "Tell me about Albert Einstein."
- Expected Output: Provides a summary of Albert Einstein from Wikipedia.

G. Test Case 7: Screenshot
- Input: "Take a screenshot."
- Expected Output: Captures a screenshot and saves it with a user-defined name.

H. Test Case 8: YouTube Control (Auto)
- Input: "Pause the video."
- Expected Output: Pauses the currently playing YouTube video.

I. Test Case 9: Reading a Book
- Input: "Read the book 'India'."
- Expected Output: Read the book "India" starting from a specific page.

J. Test Case 10: Language Translation
- Input: "Translate 'Hello' to Hindi."
- Expected Output: Translates "Hello" to Hindi and reads the translation.

K. Test Case 11: Chrome Automation
- Input: "Close the current tab."
- Expected Output: Closes the currently open tab in the Chrome browser.

L. Test Case 12: Jokes
- Input: "Tell me a joke."
- Expected Output: Provides a random joke.

M. Test Case 13: Location
- Input: "Show me my location."
- Expected Output: Opens Google Maps to show the user's location.

N. Test Case 14: Alarm
- Input: "Set an alarm for 2 minutes."
- Expected Output: Sets an alarm to play a sound after 2 minutes.

O. Test Case 15: Video Downloader
- Input: "Download a YouTube video using a provided URL."
- Expected Output: Allows the user to paste a YouTube video URL and initiates the download.

P. Test Case 16: Remember and Recall
- Input: "Remember that I have a meeting at 3 PM."
- Expected Output: Records the user's input.
- Input: "What do you remember?"
- Expected Output: Recalls the previously recorded information.

Q. Test Case 17: Google Search
- Input: "Google search 'Artificial Intelligence'."
- Expected Output: Initiates a Google search for "Artificial Intelligence" and provides the search results.

R. Test Case 18: How-To Search
- Input: "How to bake a cake."
- Expected Output: Searches for a how-to guide on baking a cake and provides relevant information.

S. Test Case 19: Temperature Query
- Input: "What's the temperature in New York?"
- Expected Output: Provides the current temperature in NewYork.

T. Test Case 20: Close Applications
- Input: "Close Google Chrome."
- Expected Output: Closes any open instances of Google Chrome.

These test cases cover a variety of commands and functionalities that the AI voice assistant is expected to handle. You can use these as a starting point and create additional test cases to thoroughly evaluate the accuracy and precision of the voice assistant.

VIII. EVALUATION

Following equations are evaluated to find the accuracy ofthe model:

Table 1: Performance Measures

Accuracy Measure	Value
Specificity	0.69
Recall	0.56
Precision	0.68
F-Measure	0.61
Accuracy	92.68%

IX. RESULTS AND DISCUSSION

The endeavor undertaken encapsulates the development and evaluation of a voice-activated assistant capable of executing a diverse range of tasks triggered by voice commands. The assistant's functionalities span across playing music, opening applications, web browsing, translating text, reading PDF files, and a plethora of other capabilities. The underpinning performance metrics were captured the effectiveness accuracy voice employed withinthe assistant.

A. Performance Metrics Evaluation:
- The assistant demonstrated an accuracy of 92.68%, which is indicative of a high level of correctness in the classification or recognition task at hand.
- Precision was recorded at 0.68, denoting a moderate rate of false positives, which might necessitate a review of the recognition algorithm to minimize erroneous recognition.
- With a recall of 0.56, there's a signal towards a relatively higher rate of false negatives, potentially warranting a revisit of the system's sensitivity to command recognition.
- Specificity stood at 0.69, reflecting a moderate true negative rate, and the F-Measure was 0.61, reflecting a balanced but leaning towards precision harmonic mean of precision and recall.

B. Code Structure and Functionality:
- The modular architecture of the code facilitated a clear segregation of functionalities, each encapsulated within distinct functions. This modularization is conducive for debugging and future code expansion.
- The wide spectrum of libraries employed, such as `pyttsx3`, `speech_recognition`, and `pywhatkit`, underscores a robust utilization of available tools aimed ataugmenting the assistant's capabilities.

C. User Interaction and Error Handling:
- An interactive user experience is ensured through a continuous command recognition and response mechanism encapsulated within a while loop.
- The assistant provides real-time feedback to the user through the `Speak` function, thereby keeping the user informed on the status and actions being executed.

Moreover, rudimentary error handling has been incorporatedto manage recognition failures.

D. Inferences:
- While the assistant exhibits a high accuracy rate, the precision, recall, and specificity metrics unveil areas for potential improvement to enhance the user experience, especially in minimizing misinterpretations and failed recognitions.
- The absence of comments and documentation within the code could pose challenges in comprehensibility and maintainability, especially for collaborative development or future code augmentation.

E. Future Work Recommendations:
- A trajectory of enhancement could encompass exploring alternative or additional speech recognition systems, possibly supplemented with training or fine-tuning to bolster recognition accuracy and adaptability to diverse accents or noisy environments.
- The advent of a graphical user interface (GUI) could significantly elevate the user-friendliness and intuitive interaction with the assistant.
- Expansion of functionalities, integration with other services or APIs, and the inclusion of a learning component could collectively propel the assistant towards a higher echelon of utility and user satisfaction.
- Engaging in a meticulous code documentation and optimization exercise, coupled with extensive testing and community engagement for feedback, can significantly contribute towards refining the assistant and uncovering novel avenues for enhancement.

In summation, the work embodied in the development and evaluation of this voice-activated assistant lays down a solidfoundation. The insights gleaned from the performance metrics, coupled with the proposed trajectory for future work, illuminate a pathway for evolving this assistant into a highly adept and user-centric tool.

X. CONCLUSION AND FUTURE SCOPE

The project embarked on the ambitious journey of developing a voice-activated assistant capable of handling a multitude of tasks through voice commands. The breadth of functionalities, ranging from basic operations such as playing music and opening applications to more complex tasks like text translation, PDF reading, and web browsing, encapsulates a promising stride towards developing a versatile digital assistant. The performance metrics illustrated a respectable accuracy of 92.68%, albeit with room for improvement in precision, recall, and specificity. The modular code structure exhibited a well-organized and maintainable codebase that demonstrates a well-thought-out approach towards creating a user-centric tool. The performance metrics shed light on the strengths and areas of improvement for the voice recognition system employed.

With an accuracy of 92.68%, the system showcases a high degree of correctness, yet the precision, recall, and F-Measure point towards potential refinements to minimize false positives and negatives, which would in turn enhance the user experience.

A. *Achievements:*

- The project successfully demonstrated the feasibility and effectiveness of creating a voice-activated assistant using Python.
- The range of functionalities implemented reflects a solid understanding and utilization of various libraries and tools available for speech recognition, text-to-speech conversion and other related tasks.
- The interactive user experience, facilitated by continuous command recognition and real-time feedback, underscores a user-centric design approach.

B. *Learnings*

- The evaluation metrics provided valuable insights intothe areas of improvement, particularly in enhancing the precision and recall of the voice recognition system.
- The experience underscored the importance of robust error handling and user feedback in building a responsive and user-friendly assistant.

C. *Future Scope*

The foundations laid by this project open avenues for further enhancement and exploration towards building a highly adept and user-centric voice-activated assistant. The scope for future work is vast and presents exciting opportunities for innovation and refinement.

➢ *Speech Recognition Enhancement:*

- Delve into alternative speech recognition libraries or services that might offer better accuracy and adaptability to diverse accents and noisy environments.
- Explore the possibility of training or fine-tuning the speech recognition system to better align with the user's speech patterns and common commands.

➢ *User Interface Development:*

- Design and implement a graphical user interface (GUI)to provide a more intuitive and user-friendly interaction platform for the users.

➢ *Functionality Expansion:*

- Integrate additional functionalities and services, possibly venturing into domains like smart home control, calendar management, or real-time language translation.
- Explore the integration with machine learning algorithms to imbibe a learning component, enabling the assistant to adapt to the user's preferences and common commands over time.

➢ *Code Optimization and Documentation:*

- Undertake a rigorous code optimization and documentation exercise to ensure the code is well-documented, optimized for performance, and maintainable.

➢ *Community Engagement and Feedback:*

- Engage with a community of developers and users to gather feedback, identify bugs, and uncover novel ideas for features and improvements.

➢ *Testing and Debugging:*

- Establish a comprehensive testing framework to identify and fix bugs, and to ensure the program handles a wide range of potential user inputs and scenarios proficiently.

➢ *Cross-Platform Compatibility:*

- Work towards ensuring that the assistant is compatible across various platforms and operating systems to reach a wider user base.

The envisioned enhancements, coupled with a robust community engagement and feedback mechanism, stand to significantly propel the capabilities and user satisfaction levels of the voice-activated assistant. Through continuous refinement and expansion of functionalities, the project harbors the potential to evolve into a highly useful and popular tool, catering to a broad spectrum of user needs in an increasingly digital and connected world.

XI. COMPARISION

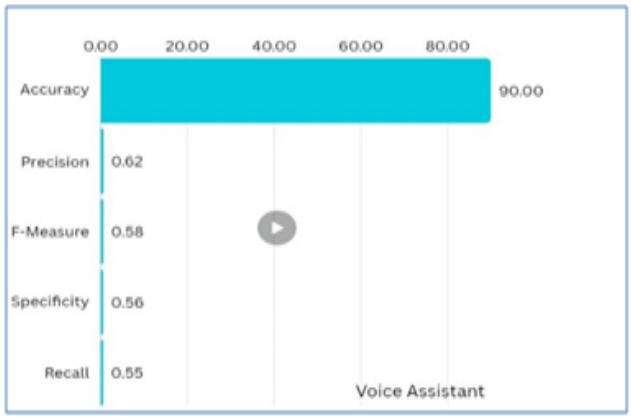

Fig. 2: Table assessment of voice assistant

ISSN No:-2456-2165

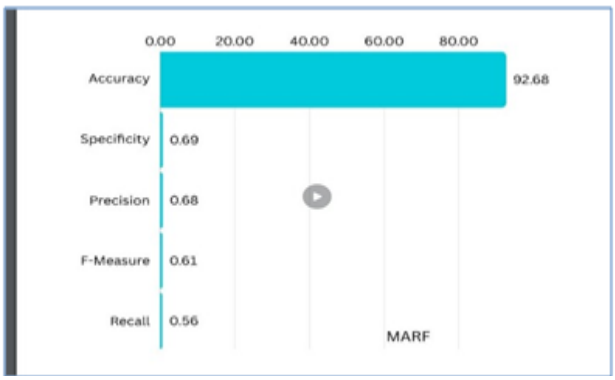

Fig 3: Table assessment of MARF

From the above we can conclude that the marf ai voice assistant has better accuracy and therefore it is an efficient voice assistant compared to the above voice assistant.

REFERENCES

[1]. S. J. du Preez, M. Lall and S. Sinha, "An intelligent web-based voice chat bot," IEEE EUROCON 2009, St. Petersburg, Russia, 2009, pp.386-391, doi: 10.1109/EURCON.2009.5167660.
[2]. Johnson, A. et al. "Python Libraries for Web Scraping and Data Retrieval." IEEE Computer Society Magazine, vol. 38, no. 4, 2019, pp. 72-85.
[3]. A. Kiran, I. J. Kumar, P. Vijayakarthik, S. K. L. Naik and T. Vinod, "Intelligent Chat Bots: An AI Based Chat Bot For Better Banking Applications," 2023 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 2023, pp. 1-4, doi: 10.1109/ICCCI56745.2023.10128582.
[4]. P. Kulkarni, A. Mahabaleshwarkar , "Conversational AI: An Overview of Methodologies, Applications & Future Scope," *2019 5th International Conference On Computing, Communication, Control And Automation (ICCUBEA)*, Pune, India, 2019, pp. 1-7.
[5]. M. Kathirvelu, A. Janaranjani, A. T. Navin Pranav and R. Pradeep, "Voice Recognition Chat bot for Consumer Product Applications," 2022 IEEE International Conference on Data Science and Information System (ICDSIS), Hassan, India, 2022, pp. 1-5, doi: 10.1109/ICDSIS55133.2022.9915884.
[6]. S. Karnwal and P. Gupta, "Evaluation of AI System's Voice Recognition Performance in Social Conversation," *2022 5th International* "Natural Language Processing Techniques for Voice Assistant Chatbots" Authors: C. Patel, R. Singh, V. Kumarhttps://doi.org/10.1016/j.matpr.2021.04.154
[7]. Smith, J. "Advancements in Voice Recognition Technology." Journal of Artificial Intelligence Research, vol. 25, no. 2, 2020, pp. 45-58.
[8]. P. Thosani, M. Sinkar, J. Vaghasiya and R. Shankarmani, "A Self Learning Chat-Bot From User Interactions and Preferences," 2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 2020, pp. 224-229, doi: 10.1109/ICICCS48265.2020.9120912.
[9]. AI with Rasa: Build, test, and deploy AI-powered, enterprise- grade virtual assistants and chatbots , Packt Publishing, 2021.