REST Journal on Emerging trends in Modelling and Manufacturing

Vol: 9(2), June 2023

REST Publisher; ISSN: 2455-4537 (Online)

Website: https://restpublisher.com/journals/jemm/

DOI: https://doi.org/10.46632/jemm/9/2/4

Desktop Based Personal Assistance Using Python Programming Language

***Ms. S. S. Kiruthika, Bhushan Kekane, Animesh Sarma, Ritul Mishra**

SRM Institute of Science and Technology, Chennai, India.

*Corresponding Author Email: kiruthis3@srmist.edu.in

Abstract: *Chat bots, conversational interfaces, and personal assistants all reimagine how people engage with computers. A virtual personal assistant allows users to ask questions in the same way they would talk to a person, and some basic actions, like opening apps, reading news, taking notes, etc., may be performed with just a voice command. Personal assistants like Google Assistant, Alexa, and Siri operate using text-to-speech and speech recognition. For instance, those who are unable to walk utilize the Internet of Things function to operate and maintain household items. So, we created a voice assistant that help users in carrying out day to day task with help of internet. Also, it does certain activities when the internet is absent. So, a voice assistant that will be practical for users and comparable to other voice assistants that are now popular has been created.*

Keywords: *Artificial Intelligence (AI), NLP, Pre-processing Python, Voice assistant.*

1. INTRODUCTION

Our lives now wouldn't be the same without personal helpers. It's because of all the features and user-friendliness they offer. A user can concentrate on what is most important to them by using a personal assistant to automate some of their daily duties. Personal assistants include functions such as making calls, sending messages, taking pictures, maintaining to-do lists when travelling, and internet browsing, among others. Therefore, using these qualities of a virtual assistant will enable someone to save a lot of time and effort. Whether it's personal or professional employment, it's critical to concentrate more on what matters most to a particular person. These kinds of personal assistants can automate mundane tasks that people frequently spend more time on. When working in an unfamiliar setting, people frequently struggle to discover the software they require, such as a browser, an IDE, or any other application. Most of the time, people will waste hours looking for the application. This wastes time that could be used elsewhere. A personal assistant will therefore aid in automating this process. The user is only required to issue a verbal command; the assistant will do the rest. By stating its name, "Dora," the virtual assistant can be called to life and will respond with a greeting appropriate to the time of day. The user's name is also requested by the assistant, which utilizes it in the welcome message. The user's commands are changed to lower case for the program's main loop to more easily recognize them. The assistant can carry out a variety of tasks, including looking up information on Wikipedia, launching Google, YouTube, and Stack Overflow websites, playing music from a local music directory, displaying the clock, sending emails, and starting programs like Opera. The code is a simple virtual assistant implementation overall, but it may be expanded to carry out more complex functions and work with other APIs and libraries.

2. DOMAIN INTRODUCTION

Artificial intelligence (AI) is the study of how to make computer systems capable of doing things like sensing, reasoning, learning, and decision-making that would typically need human intelligence. A subfield of computer science known as artificial intelligence (AI) focuses on creating algorithms and computational models that allow machines to learn from and communicate with their surroundings in a manner akin to that of a human being. A few strategies for developing AI systems are machine learning, deep learning, natural language processing, and

computer vision. Large amounts of data are used to train machine learning algorithms to detect patterns and make predictions. Deep learning, a subset of machine learning, trains neural networks with numerous layers of linked nodes. Natural language processing includes training computers to grasp and generate human language, whereas computer vision entails teaching machines to recognise and comprehend visual input. Healthcare, finance, transportation, and manufacturing are just a few of the sectors and applications that AI has the potential to revolutionize. It can be used to automate boring or repetitive operations, find trends and insights in vast amounts of data, and support decision-making. The ethical ramifications of AI, however, are also a source of concern due to factors including the possibility of biases being built into algorithms, the effect on employment, and the use of AI for surveillance or other negative objectives.

3. LITERATURE SURVEY

1. This research article suggests a voice- and gesture-based virtual assistant that both people with and without disabilities can employ to carry out routine computer operations. The major objective is to create natural human-machine contact, and the suggested solution is practical throughout the day. It can be quite useful during situations like the Covid-19 pandemic.
2. This paper talks about a virtual assistant in Bangla called Adrisya Sahayak (Invisible Helper) which was created to aid people with vision impairments in using computers, peripherals, and household appliances. It is user-independent and uses human voice instructions in Bangla to carry out activities within.
3. This study suggests a conversational agent system to offer continuous and immediate support to colleges. Conversational agents are natural language interaction interfaces designed to replicate human interactions using AI.
4. This project thesis examines the use of emerging technologies to create an intelligent Virtual Desktop Assistant with a user-based data focus. Natural language processing is employed to activate social communication skills and store data in the user's context.
5. The major task of a voice assistant prototype is to minimize the need of input devices to save hardware cost and space required. Voice assistants have been created and upgraded for greater performance and efficiency.

4. EXISTING SYSTEM

There are several existing systems for Python-based virtual assistants. Here are a few examples:

1. Jarvis AI is a Python-based open-source virtual assistant that can open webpages, play music, create reminders, and other functions. It is made to be adaptable and simple to use.
2. Another open-source virtual assistant built on Python is called Mycroft AI. It interprets user requests using natural language processing and provides an answer. It can do several things, including sending messages, creating reminders, and managing smart home appliances.
3. Rasa is an open-source, Python-based conversational AI platform. It is made to help developers create chatbots and virtual assistants that can converse with people in natural language. For several platforms, including Slack, Facebook Messenger, and others, it can be used to create virtual assistants.
4. The Python module ChatterBot can be used to create chatbots and virtual assistants. It generates responses to user inputs using machine learning techniques. It can be incorporated into many platforms, including websites and chat applications.

5. PROPOSED SYSTEM

The suggested system will be created to carry out a variety of duties, such as creating reminders, sending emails, delivering weather updates, arranging appointments, playing music, and more. Advanced NLP and ML techniques will be used by the virtual assistant to thoroughly comprehend and address user queries. As a result, the virtual assistant's precision and dependability will eventually increase. Over time, the virtual assistant will be able to pick up on the user's tastes and behaviors and adjust, giving them a more tailored experience. To give more pertinent and useful responses, it will be able to recognize and remember prior interactions and requests. The virtual assistant will be made to work with a range of technology, including wearables, smartphones, and smart home appliances. By doing this, consumers would be able to operate their gadgets using voice commands and a virtual assistant. By putting security and privacy first, the suggested system would use encryption, secure authentication, and data protection mechanisms to guarantee that user information is kept private and secure.

6. ADVANTAGES OF PROPOSED SYSTEM

The proposed system of a desktop-based personal assistant using Python programming language has several advantages over existing systems: No internet connection required: Unlike most existing personal assistants, the proposed system does not require an internet connection to function. This means that users can access the assistant's functionality even when they are offline, providing greater flexibility and convenience. Customizable: The system can be customized to meet the specific needs and preferences of individual users. Users can define their own commands, create personalized workflows, and add or remove functionality as desired. Improved accuracy: By using deep learning models for feature extraction and natural language processing, the proposed system can achieve higher levels of accuracy and effectiveness in understanding user input and providing appropriate responses. Privacy and security: Because the system does not rely on cloud-based services, it offers greater privacy and security for users who are concerned about the collection and use of their personal data. Open source: The system is built using open-source technologies, which means that users can access and modify the source code to suit their needs. This promotes greater transparency and collaboration in the development of personal assistant technologies. Offline access to information: Since the system does not rely on an internet connection, it can provide offline access to information such as the user's calendar, to-do list, and contact list. This can be particularly useful in situations where the user does not have access to an internet connection. Platform independence: The system can be run on any desktop platform that supports Python, such as Windows, macOS, and Linux. This makes it accessible to a wider range of users who may be using different operating systems. Cost-effective: The use of open-source technologies and the lack of a reliance on cloud-based services can make the proposed system more cost-effective than some existing personal assistants that require expensive subscriptions and ongoing fees. Improved workflow: By providing a central hub for managing tasks, schedules, and other information, the proposed system can help to improve the user's workflow and productivity. Easy to use: With a user-friendly interface and natural language processing capabilities, the proposed system can be easy to use for people of all skill levels. This can help to increase the adoption of personal assistant technologies by a wider range of users.

Module

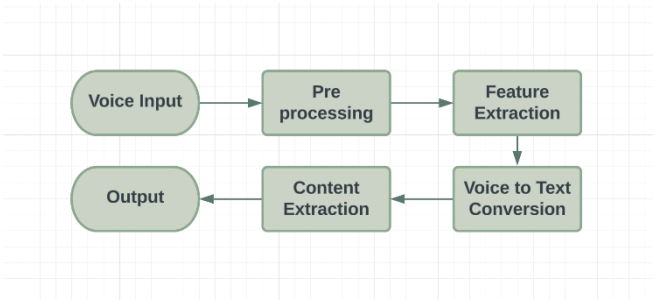

FIGURE 1. Block Diagram

Voice Input: Voice input refers to the process of capturing spoken words and sounds using a microphone or other audio input device. In the context of a desktop-based personal assistant system using Python, voice input is typically the primary mode of interaction between the user and the system. Here are the main steps involved in voice input: Audio capture: The first step is to capture the user's voice input using a microphone or other audio input device. This can be done using the operating system's built-in audio input settings or a Python library such as PyAudio. Pre-processing: The captured audio may contain noise, echoes, or other interference that can affect the accuracy of the speech recognition. Preprocessing techniques, such as filtering, noise reduction, and normalization, can be used to improve the quality of the audio. Speech recognition: The captured audio is then passed through a speech recognition system, which uses algorithms and machine learning models to convert the spoken words into text. Text analysis: The resulting text is then analyzed by the system to identify the user's intent and extract relevant information, such as keywords, entities, and context. Response generation: Based on the text analysis, the system generates a response, which can be in the form of spoken words, text, or other output. Voice input is a key component of many personal assistant systems, such as Apple's Siri, Amazon's Alexa, and Google Assistant. In recent years, advances in speech recognition and natural language processing technologies have made it possible for these systems to accurately recognize and respond to user input in a more natural and intuitive way.

Pre-processing: In the context of a desktop-based personal assistant system using Python, preprocessing refers to the set of techniques used to prepare and clean up the raw data, in this case the user's voice input, before it is analyzed by the system. Here are some common preprocessing techniques used in speech processing: Signal filtering: This involves removing any noise or unwanted frequencies from the audio signal. Common filtering techniques include high-pass, low-pass, and band-pass filters. Noise reduction: This technique involves removing background noise and interference from the audio signal. Common noise reduction techniques include spectral subtraction, Wiener filtering, and signal thresholding. Normalization: This technique involves adjusting the amplitude or volume of the audio signal to a standard level. This can be done by scaling the signal to a fixed maximum amplitude or by adjusting the gain. Resampling: This involves changing the sampling rate of the audio signal, which can be useful in cases where the input device has a different sampling rate than the processing system. Feature extraction: This involves identifying and extracting relevant features from the audio signal, such as pitch, formants, and spectral characteristics. These features can then be used as input to the speech recognition system. Preprocessing is an important step in speech processing, as it can significantly improve the accuracy and quality of the speech recognition output. There are several Python libraries that can be used for speech preprocessing, including PyAudio, SciPy, and Librosa.

Feature Extraction: In the context of a desktop-based personal assistant system using Python, feature extraction refers to the process of identifying and extracting relevant features from the preprocessed audio signal, in order to provide input to the speech recognition system. The goal of feature extraction is to represent the audio signal in a more compact and meaningful way, by identifying the key characteristics that are relevant for speech recognition. Here are some common techniques used for feature extraction in speech processing:

1. Mel-frequency cepstral coefficients (MFCCs): This is a widely used technique for speech feature extraction, which involves applying a filter bank to the preprocessed audio signal and then computing the logarithm of the resulting power spectra. The resulting features are then transformed using a discrete cosine transform (DCT) to obtain the MFCCs.
2. Linear predictive coding (LPC): This is another technique for speech feature extraction, which involves modeling the speech signal as a linear combination of past and present speech samples. The resulting LPC coefficients can be used as input to the speech recognition system.
3. Perceptual linear prediction (PLP): This is a variant of LPC that takes into account the non-linear properties of the human auditory system. PLP coefficients can provide a better representation of the speech signal in noisy environments.
4. Pitch estimation: This involves estimating the fundamental frequency of the speech signal, which can be a useful feature for speech recognition. Pitch estimation can be done using techniques such as autocorrelation, harmonic product spectrum, or the YIN algorithm.
5. Spectral features: This involves extracting spectral characteristics of the speech signal, such as formants, spectral peaks, and spectral moments. These features can be useful for discriminating between different phonemes and words.

Feature extraction is an important step in speech processing, as it can significantly improve the accuracy and efficiency of the speech recognition system. There are several Python libraries that can be used for speech feature extraction, including Librosa, SciPy, and Python_speech_features.

Voice to Text Conversion: In the context of a desktop-based personal assistant system using Python, voice-to-text conversion refers to the process of transcribing the preprocessed and feature-extracted audio signal into text that can be used as input for further processing. Here are some common techniques used for voice-to-text conversion in speech processing:

1. Hidden Markov Models (HMMs): This is a widely used technique for speech recognition, which involves modeling the relationship between the audio signal and the corresponding text using a statistical model based on Markov chains. HMMs can be trained on a large corpus of speech data to recognize different words and phrases.
2. Deep neural networks (DNNs): This is another technique for speech recognition, which involves training a neural network to recognize the features extracted from the preprocessed audio signal. DNNs have been shown to achieve state-of-the-art performance in speech recognition tasks.
3. Convolutional neural networks (CNNs): This is a variant of DNNs that is particularly suited for processing time-series data, such as audio signals. CNNs can be used to extract features directly from the raw audio signal, which can improve the accuracy of the speech recognition system.
4. Recurrent neural networks (RNNs): This is another variant of DNNs that is particularly suited for processing sequential data. RNNs can be used to model the temporal dependencies in the audio signal, which can improve the accuracy of the speech recognition system.
5. Speech recognition libraries: There are several Python libraries that provide pre-trained models for voice-

to-text conversion, such as Google Speech Recognition, CMU Sphinx, and Kaldi. These libraries can be used to quickly integrate speech recognition into a desktop-based personal assistant system.

Voice-to-text conversion is a crucial step in a desktop-based personal assistant system, as it allows the system to understand and process the user's voice input. There are several Python libraries and techniques that can be used for voice-to-text conversion, depending on the specific requirements of the system.

Content Extraction: In the context of a desktop-based personal assistant system using Python, content extraction refers to the process of extracting relevant information from text input, such as user queries or web pages. The goal of content extraction is to identify and extract the key pieces of information that are relevant to the user's query or task. Here are some common techniques used for content extraction in text processing:

1. Named entity recognition (NER): This is a technique for identifying and extracting named entities, such as people, organizations, and locations, from text. NER can be used to extract information such as names, dates, and locations from user queries or web pages.
2. Part-of-speech (POS) tagging: This is a method for recognising and labelling sentence components of speech such as nouns, verbs, and adjectives. The important concepts and themes in a user query or web page can be identified via POS tagging.
3. Dependency parsing: This is a way for evaluating the grammatical structure of a phrase and discovering the relationships between distinct elements. Dependency parsing can help you understand what the user is saying by determining the subject, object, and verb of a sentence.
4. Sentiment analysis: This is a strategy for determining if a piece of literature has a good, negative, or neutral emotional tone. Sentiment analysis may be used to determine a user's emotional state or opinion on a specific issue.
5. Text summarization: This is a technique for generating a summary of a longer piece of text, such as a web page or document. Text summarization can be used to provide a concise and relevant summary of the key information in a user query or web page.

Content extraction is an important step in text processing, as it allows the desktop-based personal assistant system to understand and process the user's text input. There are several Python libraries that can be used for content extraction, including NLTK, Spacy, and TextBlob. The specific techniques and libraries used will depend on the requirements of the personal assistant system and the type of information being extracted.

Output: The output of a desktop-based personal assistant system using Python will depend on the specific functionality and features of the system. However, some common output types include: Text output: This can include text-based responses to user queries or commands, such as weather forecasts, news updates, or reminders.

1. Voice output: This can include spoken responses to user queries or commands, which can be generated using text-to-speech (TTS) technology.
2. Graphical output: This can include visualizations or graphs that display information in a more intuitive and user-friendly way, such as stock market trends or weather forecasts.
3. Notification output: This can include notifications or alerts that notify the user of important events or reminders, such as upcoming appointments or deadlines.
4. Action output: This can include executing actions or tasks based on user commands, such as sending emails, scheduling appointments, or setting reminders.

The output of a desktop-based personal assistant system should be designed to be user-friendly, intuitive and should provide relevant and accurate information based on the user's query or command. The specific types of output used will depend on the requirements and goals of the personal assistant system, as well as the preferences of the user.

7. RESULT AND DISCUSSION

The result and discussion of a desktop-based personal assistant system using Python will depend on the specific functionality and features of the system, as well as the evaluation metrics used to measure its performance. Here are some possible results and discussions that could be relevant for such a system:

1. Accuracy: One key metric for evaluating the performance of a personal assistant system is accuracy. This can be measured in various ways, depending on the specific tasks and functions of the system. For example, for voice recognition, accuracy can be measured in terms of the percentage of words correctly recognized. For content extraction, accuracy can be measured in terms of the percentage of key pieces of information correctly identified and extracted.

2. Speed: Another important metric for a personal assistant system is speed, or the time it takes for the system to respond to user queries or commands. Speed can be measured in terms of response time, or the time it takes for the system to generate a response to a user query or command. Faster response times can improve the user experience and make the system more efficient.
3. User satisfaction: The goal of a personal assistant system is to improve the user's experience and provide useful and relevant information. Therefore, user satisfaction is an important metric for evaluating the performance of the system. User satisfaction can be measured through user feedback, such as surveys or ratings of the system's performance.
4. Customization: Another important aspect of a personal assistant system is customization, or the ability to tailor the system to the user's preferences and needs. Customization can be measured in terms of the degree to which the system can be customized, such as the ability to personalize responses, adjust settings, or integrate with other applications.

The result and discussion of a desktop-based personal assistant system using Python should be focused on the system's performance, effectiveness, and user satisfaction. The system should be evaluated on key metrics that reflect its ability to understand and respond to user queries or commands, and to provide relevant and useful information in a timely and efficient manner. The discussion should also consider the potential limitations and challenges of the system, and ways to address these challenges to improve the system's performance and usability.

8. CONCLUSION

Users could benefit from a practical and understandable tool for managing activities, getting to information, and more using a desktop-based personal assistant system that utilizes Python. Enhancing productivity and providing information. Voice input, preprocessing, feature extraction, voice-to-text conversion, content extraction, and output are some of its crucial elements. Metrics like accuracy, speed, user happiness, and customizability can be used to gauge the system's success, and the potential rewards are substantial. For individuals interested in creating applications for artificial intelligence and natural language processing, it is worthwhile to investigate.

REFERENCES

[1] Tata Jagannadha Swamy; M Nandini; Nandini B; Venkata Karthika K; V Laxmi Anvitha; Ch Sunitha (2021), “Voice and Gesture based Virtual Desktop Assistant for Physically Challenged People”, 6th International Conference on Trends in Electronics and Informatics (ICOEI) **DOI:** 10.1109/ICOEI53556.2022.9776746.

[2] Md. Rakibuz Sultan; Md Moinul Hoque; Farah Ulfath Heeya; Iftiquar Ahmed; Md. Redwanul Ferdouse; Shikder Mejbah Ahmed mubin (2022), "Adrisya Sahayak: A Bangla Virtual Assistant for Visually Impaired", 2nd International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST) **DOI:** 10.1109/ICREST51555.2021.9331080

[3] Mehdi Mekni (2021), " An Artificial Intelligence Based Virtual Assistant Using Conversational Agents”, Journal of Software Engineering and Applications DOI: 10.4236/jsea.2021.149027

[4] Zeeshan Raza, Aftab Amin Sheikh, Ankush Shahu, Abhijeet Thakur (2022), “VIRTUAL DESKTOP ASSISTANT USING PYTHON”, International Research Journal of Modernization in Engineering Technology and Science.

[5] Raj Kumar Jain, Vikas Sharma, Mangilal, Rakesh Kardam,Mamta Rani (2021), “Artificial Intelligence Based A Communicative Virtual Voice Assistant Using Python & Visual Code Technology” World Journal of Research and Review (WJRR)

