ROBOTA: THE AI VIRTUAL ASSISTANT

P Selvaraj, Bhavya Singhal, Kushagra Sharma

Department of Computer Science, Galgotias University, Greater Noida

KEYWORDS

Text – to - speech
Voice Control
Visually Impaired
Speech Recognition

ABSTRACT

It has been found through research that visually impaired people face a lot of difficulties while accessing the Internet compared to people with other disabilities. The number of visually impaired users is so large that most of them have to depend on the help of others to complete any task on the Internet. This paper introduces software that enables visually impaired people to do various Internet-related activities independently. This software will enable them to surf the Internet as smoothly as any other normal user. Sending messages, calling, or going to YouTube-everything is a big headache for a blind user in this fast-moving world. This software overcomes such challenges by enabling the visually impaired to do these types of things with voice commands to turn on Assistant or keys on Braille keyboards. This input, therefore, goes through the assistant, which then performs the required operations and gives output to the user. It uses a speech-to-text recognition module to understand commands and execute tasks for revising results. In sending text, for instance, it could use the database to check whether the contact is available and whether the message has been sent. It is also available for all seeing users via a text input field. It can revolutionize the use of the Internet for visually impaired users and greatly extend their abilities in acting within the digital world.

1 Introduction

WHO estimates that 2.1 billion people currently have near or distance vision impairment, when at least 1 billion could also have been prevented by treatment that are yet to get it, in which the percentage keeps growing to date due to rising aging population. [1]. To them, it is the "Internet" that brings all those open avenues for communication, education, and employment opportunities. At a time when the digital technologies were growing phenomenally, the visually impaired face immense difficulties in accessing the net. Everything can be done online now-from watching a video to ordering food to messaging someone online. For almost everything online, facilities that a person needs require the use of the Internet. While using the Internet may be a somewhat trivial task for most people, this has been quite a challenge for the blind and visually impaired. Thus, we wanted to find some unique way of allowing Visually impaired people to access the internet. The Internet is a very visual form of various types of websites may offer different accessibility barriers, as compared to the others where accessibility could be guaranteed by putting in a ramp Wheelchairs or Braille interfaces.

We overcome this limitation by underlining exactly where visually impaired users differ from regular users concerning problems they experience using the Internet. In this paper, we presented a virtual assistant software that lets the blind would not need to learn keyboard shortcuts. On the other hand, it can be used manually by simply typing in an input field by the normal user. It can be initiated with a wake word or simply by using the keyboard, adding a query directly into the input field. The software uses a speech-to-text module, which helps in converting the input voice to text and proceeds with the problem. It lets the user know what action they are doing once the command has been given to the software. For example, if the user says, "Open YouTube," then the software replies whether YouTube has been opened or not. The same thing happens with everything else.

The American Foundation for the Blind [2] estimated that people with visual impairments are over 31% less likely to report going online and over 35% less likely to use a desktop computer than people without disabilities. Though various solutions exist today, such as screen readers or magnification tools, these have their own limitations. Most screen readers, for example, require users to commit complex keyboard shortcuts to memory. Moreover, they may not work properly for non-standard web interfaces or for websites whose design is updated often, which would then create usability problems. The gap between accessibility guidelines and the real world makes it difficult for visually impaired people to surf the internet independently.

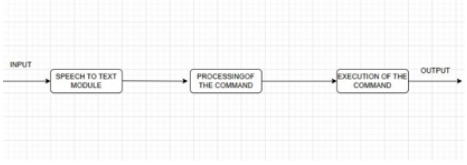

Figure 1: Flow Diagram of Solution

Figure 1 illustrates how our software works. The input speech is identified through the speech-to-text module and then further processing for the accomplishment of the tasks to provide the required output.

The big challenge in this project is to enable voice control to let users-even visually impaired or normal-enable performance. Performance is also based on the OS the user operates on. Implementation of software modules is presented that automates the most frequently used applications/websites by the users, namely YouTube, WhatsApp, any system application, and anything you ask from AI. By this, we aim to cater to the maximum possible needs of the user.

2 Literature Review

For example, Muller et al. [5] reported economic and technical capabilities as major issues of the internet's accessibility to visually impaired users. Similarly, Kirsty et al. [6] have reported that these problems are exacerbated further by poorly written HTML code and reliance on PDFs despite W3C guidelines. Pilling et al. [3] researched whether the internet acts as an enabling resource for the disabled or an extension of social exclusion; they concluded that without assistive tools, this resource is less than ideal. Sinks and Kings [4] indicated that few studies are conducted to find out what the reasons are for which disabled persons cannot use the internet like others and called for directed studies.

Power et al. [7] discovered that only 50.4% of issues encountered by users were covered by WCAG 2.0 Success Criteria, and even then, guidelines, when implemented, often did not solve important problems: for example, one of the respondents in the study of Pilling et al. [3] pointed out, "Without the software, there is no access for blind people," emphasizing dependence on special tools like JAWS. On the other hand, Ferati et al. [10] believe that even JAWS cannot provide personalized support for people suffering from different extents of visual losses. The authors would suggest adopting modular, plugin-based systems in reducing direct keyboard interactions but with greater ease of accessibility. Lastly, Porter [8] pointed out the empowerment brought about by the internet to people with visual impairment in selecting the materials themselves instead of prepared materials, which is what happens when Braille newspapers are used.

3 System Overview and Design

It basically provides four important functions. It can open system applications using a database of installed applications, and using the command prompt in the case of pre-installed applications. Opening websites is done efficiently using a database of major websites to load them quickly. Similarly, applications like WhatsApp would use the database, too, since such applications are installed by users themselves based on their needs. For the rest of the tasks, it has an integrated AI chatbot working with the cookies provided by the open-source community.

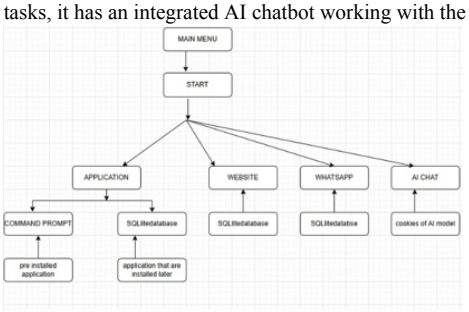

Figure 2: System Architecture

Figure 2 presents so-called system architecture of our software. User comes to the software via main menu and starts virtual assistant there. Then speech-to-text module will translate his speech into text format. Then it will be told to the user if his request will be processed or not. The model runs in response to a request by the user, through the functionality that is captured by Figure 2. This will send output back to the user via a Text-to-Speech module; this is an overview of software.

4 Methodology

A. main.js

The main.js file defines the functionality for the virtual assistant's interface in JavaScript. It handles text animations, including the "bounce-in" effect using Textillate.js, and basic fade-in/fade-out effects. Additionally, it provides a dynamic wave animation for audio input and response through SiriWave. The file manages user interactions with various buttons, such as the MicBtn for activating the microphone and sending audio commands via eel, or the send button for submitting text input. It also handles event toggling to show or hide UI elements, ensuring smooth transitions between input fields and buttons for a seamless user experience. Moreover, it listens for specific keyboard shortcuts to further enhance the user's interaction with the assistant.

B. index.html

This is the main HTML file for the project, giving the basic skeleton needed to build the web interface. This template includes Bootstrap for responsive design and layout, jQuery for DOM manipulation, and several animation and visualization libraries. It provides an appealing input interface with buttons to activate the microphone, send messages, and settings. It has a canvas inside it for graphical effects like particle animation and also includes the SiriWave.js library for showing a dynamic waving waveform. In order to provide more interaction and style with it, it also has a number of CSS and JavaScript dependencies.

C. script.js

The script.js file defines the functionality and visualization for a 3D particle animation to be rendered on an HTML5 canvas. It realizes a 3D spinning sphere by the application of mathematical transformations, which define every point in 3D Cartesian space where a point should be displayed. Global variables are declared at the top, which will be used to control parameters later on, such as the properties of the particles: position, velocity, alpha, and acceleration in general. It runs in a timed loop with setInterval; it updates the positions of all the particles and draws them with depth effects. Field of view blur is also simulated with alpha blending to further enhance the realism.

D. controller.js

The controller.js file mediates between the user interface and the back-end of the controller. It provides asynchronous communication with the server, including fetching data, sending commands, and receiving responses. It uses AJAX calls and WebSocket connections to provide real-time updates with least latency. Further, it may

include features dealing with network interruptions gracefully, for example, by retrying requests or showing informative error messages in case retries failed. It also supports progressive state management through which the frontend can maintain user preferences and context across interactions. So, for example, controller.js does support transitions from text input to voice input and vice versa while maintaining session data in between.

E. style.css

In style.css, making the virtual assistant more attractive to both user and modern outlooks is supposed to be styled. It designates the visual of backgrounds, buttons, input fields, and various animations using Gradients, Shadows, Transitions, or other CSS-based designing methods. This file edits button styles and hover effects while adapting different window size issues in the settings of input areas and chat area disposition. Furthermore, it realizes animation effects, making transitions smooth and providing interactive feedback that helps in enhancing the aesthetic and functional aspects of the application.

F. features.py

The script feature.py has some very key functionalities. This script will interface with the OS for features that involve media player features, looking up contact names, and handling conversations of the chatbot. It then interfaces with the database for getting information about which application or websites it opens and for launching specific programs or YouTube videos. The hotword detection for words like "Jarvis" or "Alexa," the support of voice command through PyAudio, does audio processing in real time. It allows users to search for contacts and send messages or call or video call them through WhatsApp. For user queries, these respond to chats using the HugChat framework.

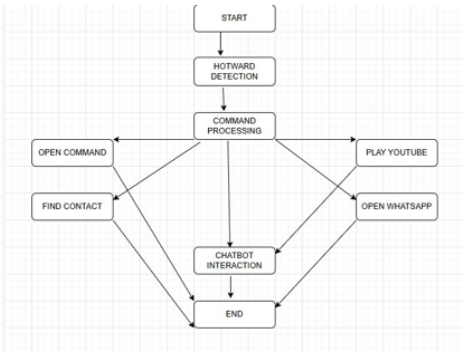

Figure 3: Working of features.py module

G. command.py

The command.py script is very important for handling Voice and Text commands. It uses pyttsx3 for text-to-speech and speech_recognition for speech-to-text. This script speaks text and also sends it to the frontend using the eel library, allowing Python to communicate with JavaScript. The takeCommand basically listens for audio input via the microphone, processes the audio through Google's Speech Recognition API, and returns text that the program recognizes. Another exposed function within the script processes commands; this is how it provides the variable allCommands to handle different tasks. These are voice commands given by the user or coming from the frontend to perform the task or action, like opening an application, YouTube video, sending WhatsApp messages or calls, and video calls. The script does include a feature of a query for a chatbot in case a user request is complex; it also treats errors to exit gracefully in case something goes wrong. This contains the eel.DisplayMessage and eel.ShowHood functions, enabling the interaction and feedback with the user.

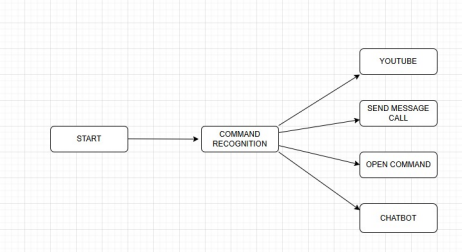

Figure 4: Working of command.py module

5 Result

Text-to-speech synthesis is performed by a Python program using the Speech Synthesis module called pyttsx3, which is integrated with Google's Speech Recognition library. The integration provides remarkable accuracy and is a very fast and efficient technique of converting text to speech. The recognition rate of the text-to-speech is about 96.25% based on 4 different speech samples and 20 different inputs, all tested in moderate to quiet environments.

The results indicated that our software can handle popular tasks such as YouTube, WhatsApp, AI Chatbot, and launching applications. Each of these functions was tested independently with the software. The software can play any video on YouTube with user controls. It also responds to whatever question a user may have with the use of a chatbot. It can even send WhatsApp messages via the software; a tool can, therefore, be developed which helps a visually impaired user to easily and efficiently access the internet.

6 Conclusion

It suggests a framework for virtual assistants that comprises modular backend functionality integrated with a comfortable-to-operate frontend. The proposed system will solve critical problems of scalability, adaptability, and user satisfaction by integrating state-of-the-art robust database management with seamless application logic. It is designed modularly to ensure it is easy to maintain and upgrade as new technologies unfold, thus keeping the framework well into the future. Experimental deployments of the framework have already shown significant gains in efficiency, scalability, and reliability. This work is presented as an indication of how the use of emotionally intelligent interactions can be explored with federated learning to advance privacy in training sets and extend multimodal capabilities to gestures and facial recognition. It would be possible to further develop this framework

toward more personalized and interactive user experiences with intelligent virtual assistants.

7 Future Enhancement

This may be subject to numerous changes in future updates, including various languages. Planned features include Face Recognition and the ability to open sites with summaries, among many others. Other development will continue regarding video and educational content being made available in a way for visually disabled users to consume information they want to as properly and fully as fully sighted users do.

8 References

[1] Global data on visual impairments 2010 by World Health Organization (WHO) - https://www.who.int/blindness/GLOBALDATAFINALforweb.pdf?ua=1
[2] The website for American foundation for the blind https://www.afb.org/about-afb/what-we-do/afb-consulting/afbaccessibility-resources/challenges-web-accessibility accessed in April 2020
[3] Pilling, D., Barrett, P. and Floyd, M. (2004). Disabled people and the Internet: experiences, barriers and opportunities. York, UK: Joseph Rowntree Foundation, unpublished.
[4] Sinks, S., & King, J. (1998). Adults with disabilities: Perceived barriers that prevent Internet access. Paper presented at the CSUN 1998 Conference, Los Angeles, March. Retrieved January 24, 2000 from the World Wide Web.
[5] Muller, M. J., Wharton, C., McIver, W. J. (Jr.), & Laux, L. (1997). Toward an HCI research and practice agenda based on human needs and social responsibility. Conference on Human Actors in Computing Systems. Atlanta, Georgia, 22–27 March.
[6] Kirsty Williamson, Steve Wright, Don Schauder, Amanda Bow, The internet for the blind and visually impaired, Journal of Computer Mediated Communication, Volume 7, Issue 1, 1 October 2001, JCMC712
[7] Power, C., Freire, A.P., Petrie, H., Swallow, D.: Guidelines are only half of the story: accessibility problems encountered by blind users on the web. In: CHI 2012, Austin, Texas USA, 5–10 May 2012, pp. 1–10 (2012)
[8] Porter, P. (1997) 'The reading washing machine', Vine, Vol. 106, pp. 34–7
[9] JAWS-https://www.freedomscientific.com/products/software/jaws/accessed in April 2020
[10] Ferati, Mexhid & Vogel, Bahtijar & Kurti, Arianit & Raufi, Bujar & Astals, David. (2016). Web accessibility for visually impaired people: requirements and design issues. 9312. 79-96. 10.1007/978-3-319-45916-5_6
[11] Ryle Zhou, Question answering models for SQuAD 2.0, Stanford University, unpublished.