Utilizing Automatic Speech Recognition to Create a Desktop Assistant

Grant Proposal

Adel Benchemam

Massachusetts Academy of Math and Science at WPI

85 Prescott Street, Worcester, Massachusetts

Author Note

I would like to thank Dr. Crowthers, Mrs. Taricco, Ila Chakravarthy, Andrew Brown, Anshu Adiga, Avani Jain, Jasmin Bella, and my mentor Alexander Galvan for guiding and giving me recommendations throughout this project.

Executive Summary

Automatic Speech Recognition (ASR) is a subfield within Artificial Intelligence (AI) commonly used for speech recognition services such as automatic captions, translations, and virtual assistants. Virtual Assistants increase convenience when using devices such as phones and computers as they can open applications, send text messages, and answer questions. However, improvements can be made to modern assistants to help people with disabilities that hinder them from being able to interact with their computers. Because of this problem, a voice assistant that can interact with apps and websites was created to allow for voice-controlled navigation of their computer. Using various Python libraries, website automation services, and app automation services, a voice assistant was created to improve the navigation and intractability of computers. Google's speech recognition service was used as an automatic speech recognition model to transcribe commands. Using these commands, automation services were used to simulate interaction with an app or website. Data was collected on the speech recognition model in order to determine its transcription accuracy under various background noise levels. With this data, a 1-proportion Z interval was used to determine a confidence interval and predict the confidence of the assistant working under those noisy conditions. The results show that the assistant is usable under background noise levels up to 60 decibels. This application will allow people with disabilities to navigate their computers through virtual assistance which is crucial as there are more opportunities for learning, entertainment, and work available through technology every single day.

Keywords: Automatic Speech Recognition, Voice Assistant, Voice Recognition, Natural Language Processing, Desktop Assistant.

Utilizing Automatic Speech Recognition to Create a Desktop Assistant

Artificial Intelligence (AI) is used everywhere in daily life, the medical field, and other sciences (Yip et al., 2023; Tinao & Jamisola, 2023). Automatic Speech Recognition (ASR) is a subfield within Artificial Intelligence commonly used for transcription services, virtual assistants, and translation devices. Since technology is always evolving, and the way we interact with it is also changing. Voice is the easiest and most effective way of communicating with technology. Voice commands led to the concept of virtual assistants, an auditory way to interact with technology. We already rely on virtual assistants to turn on and off the lights in a room, stream music, or navigate as search engines (Subhash et al., 2020). The features that a virtual assistant can have are shown in Figure 1, including tools such as list creation, social media support, email support, data entries, and research tools.

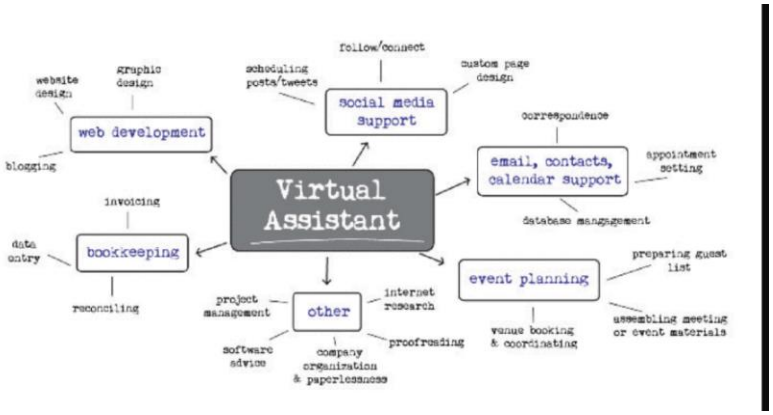


Figure 1: Virtual Assistant Features (Bajpai et al., 2024)

How does Automatic Speech Recognition work?

ASR is the driving principle behind speech recognition. It complements another component, Natural Language Processing (NLP), which allows virtual assistants to comprehend both the meaning of what a user has stated and also to infer implicit nuances (Bajpai et al., 2024). Automatic Speech Recognition systems initially record speech and save the speech as a file, the file will then be cleaned of any background noise and analyze what is stated sequentially. Natural Language Processing creates a

better contextual understanding of user queries through semantic analysis. Probability tests are applied to recognize all the words that complete the input. Finally, it will produce an output in the form of text content (Subhash et al., 2020). The result of the ASR process can be shown in Figure 2.

Figure 2: The process of ASR (Subhash et al., 2020).

Past Uses of Automatic Speech Recognition

Virtual Voice Assistants are a common application of Automatic Speech Recognition. Most people have used Apple Siri ©, Amazon's Alexa ©, Microsoft Cortana ©, or Google Assistant© to search for information, send text messages, or play music. A great application of voice assistants can be for those with restricted movements that make it difficult to use technology. Verbal communication with a virtual assistant can accomplish tasks without having to touch the device. Voice Assistants are a way to make our lives more convenient by reducing the time it takes to complete tasks. Today, Voice Assistants are integrated into many devices, such as cell phones and computers, and are available to the general public. Some assistants are hardware-based and made to do one thing, like Alexa's wall clock, and others are software-based, like the assistants someone would find in a phone (Singh et al., 2022).

While they are great for retrieving information, there is still room for improvement when it comes to user control with the device. Common voice assistants, like Siri, often contain simple app and web interaction features that can retrieve information like searches. However, they lack giving users more intractability with websites. This kind of feature can be significantly helpful to those who have limited arm mobility.

Tools

The goal of this project is to fix weaknesses that are found in Voice Assistants. These weaknesses can be fixed using a programming language named Python, which has large amounts of resources and libraries that can be utilized for this project. The audience of this software can be anyone; however, it is meant to aid those with restricted mobility. The program is split into two main components: speech recognition and local interaction. The first component uses the speech recognition Python library. This library allows someone to speak and return the content in text form. It uses the same concept of Automatic Speech Recognition and helps identify commands. The second component is more complicated as there are many forms of local interaction. The program uses the Python libraries WebBrowser and PyAutoGui for tab control and key bind access. It also uses Selenium Web Driver to interact with web elements. At times, the program will speak to the user. Communication with the user is done using the pyttsx3 library. The input can be a standard user microphone, and the output can be a standard speaker.

Future Steps

This concept can be applied to current Virtual Assistants. It can also be improved by creating a more accurate Automatic Speech Recognition model, as the current one that is being used struggles with recognizing speech in noisy environments (Dilmegani, 2024). An approach that can be done to address this issue would be to train the ASR model through Spectrum Matched Training as it shows higher accuracies in noisy environments (Prodeus & Kukharicheva, 2016). Another step can be greater application intractability for local applications. Cross-platform file searches can also be implemented, as the current program has only been tested through a Windows operating system. Another feature that can be implemented into the program is an intention detection system that will trigger the assistant when it feels that it is needed by detecting if the user is requesting the assistant based on the volume of the request relative to other sentences in the conversation surrounding it (Barton, 2015).

Section II: Specific Aims

This proposal's objective is to create a voice assistant system that will allow for more interactivity on web pages. Website interaction will be done using a variety of Python libraries that will identify HTML elements. Previous assistants such as Microsoft's Cortana, Google Assistant, and Apple's Siri don't offer features that allow them to interact with web pages through voice commands. While these assistants are very powerful and can act as a search engine and retrieve information such as weather (Jain et al., 2021), they lack intractability when it comes to websites. For example, a person cannot use these assistants to click buttons or text boxes displayed on websites.

Specific Aim 1: The program will be able to interact with website elements such as click buttons, type in textboxes, and scroll through a website.

Many large assistants lack this, and since many social and business interactions are moving to the Internet, this is necessary to help further computer interactions. People with disabilities who are not able to navigate using a mouse can use their voice to navigate websites for entertainment, work, communication, or shopping.

Specific Aim 2: The program will be cross-platform.

Assistants aren't cross-platform, as they are owned by different companies. Different company ownership means that beneficial aspects of one assistant sometimes might not be incorporated into another assistant. A critical criterion for widespread use would be cross-platform compatibility.

Specific Aim 3: The program will also have some tools that can help with selecting files for online submissions.

File sharing is another key criterion that is useful for everyday life. Although current assistants do not include file-sharing features, some features have been implemented by the ARIA bot (Singh et al., 2022), and we will take a similar approach.

The expected outcome of this work is to build upon current tools for speech recognition and web browser automation to assist people in using websites and provide for the flaws of current voice assistants.

Section III: Project Goals and Methodology

Relevance/Significance

The project is important because using technology is an increasing need, and it can be hard for some people to access. We use technology for everything, such as entertainment, learning, work, shopping, and communication. As more services and businesses move to the internet, accessibility becomes increasingly important. A voice assistant can facilitate accessibility to retrieve information and help with communication (Appalaraju et al., 2022; Kunekar et al., 2023). Through voice recognition, this project intends to transcribe user speech and pick out the commands given through ASR. Then, a variety of Python modules would perform the tasks.

Innovation

This project connects multiple tools that are not meant for this specific result but make them work together to accomplish the specific aims. It utilizes Google speech-to-text as an automatic speech recognizer for transcription, python libraries for website automation, and modules for GUI creation and control.

Methodology

The data that will be collected to test the accuracy in voice recognition will be Text to Speech voices that will read out commands (NaturalReaders, 2019). Then, transcription accuracy will be determined using the correct input and compare it to the speech recognition output and mark it as a failure or a success. A similar trial will also be done to test the functionality of the app for task completion. Afterwards, a Word Error Rate (WER) test will be performed to test the error rate the

speech recognition model has on large pieces of text. Word Error Rate is calculated as follows (Chiusano, 2022):

$$WER = \frac{S+D+I}{T} \quad (1)$$

Where S is the substitutions made to the input, D is the deletions from the input, and I is insertions to the input. The total words are represented by T (Equation 1).

Specific Aim #1: Web Interaction

The objective is to make our assistant able to interact with web elements such as buttons, text boxes, etc. Our approach is to use the Selenium library and the transcription of the user's speech. First, the program will look for the command it is given. With the transcription of the user, it will end up checking if the website has text that corresponds to what the user has said. If the text matches, then it will click it. This approach is similar to the one shown in Figure 1. This feature is not available on large assistants such as Siri or Cortana.

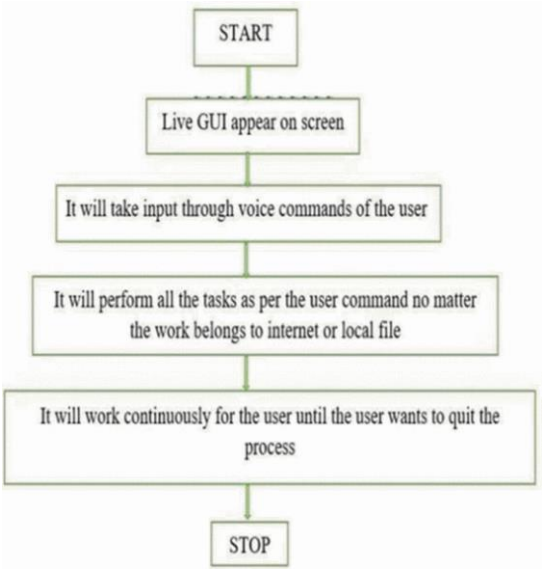


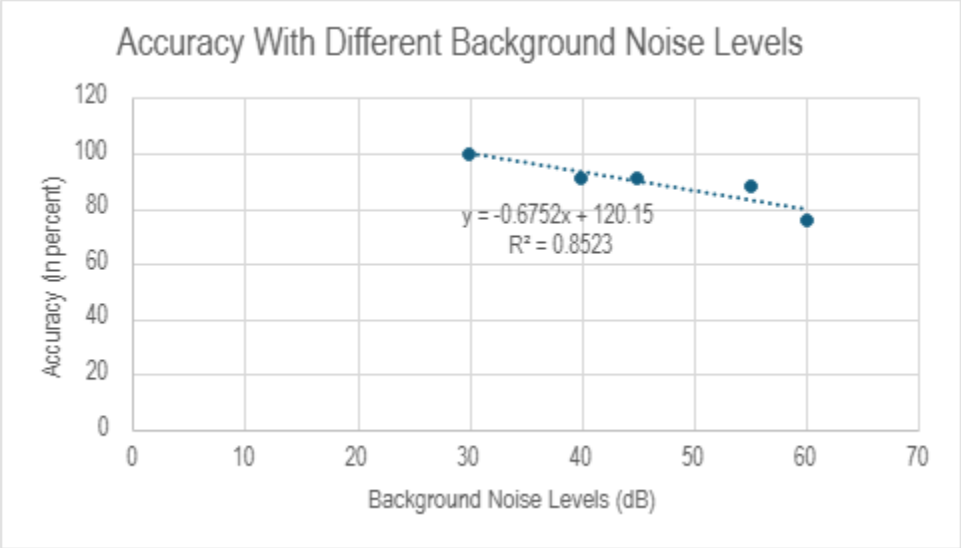
Figure 3: Graphical representation of the Assistant (Singh et al., 2022)

Justification and Feasibility: Utilizing Selenium is what makes web intractability possible. Selenium was originally a website automation tool (Selenium, n.d.). However, as the user gives it a task, the program sends in a request according to the command, such as scroll or click. Selenium then performs the task, and it makes interacting with website elements very easy.

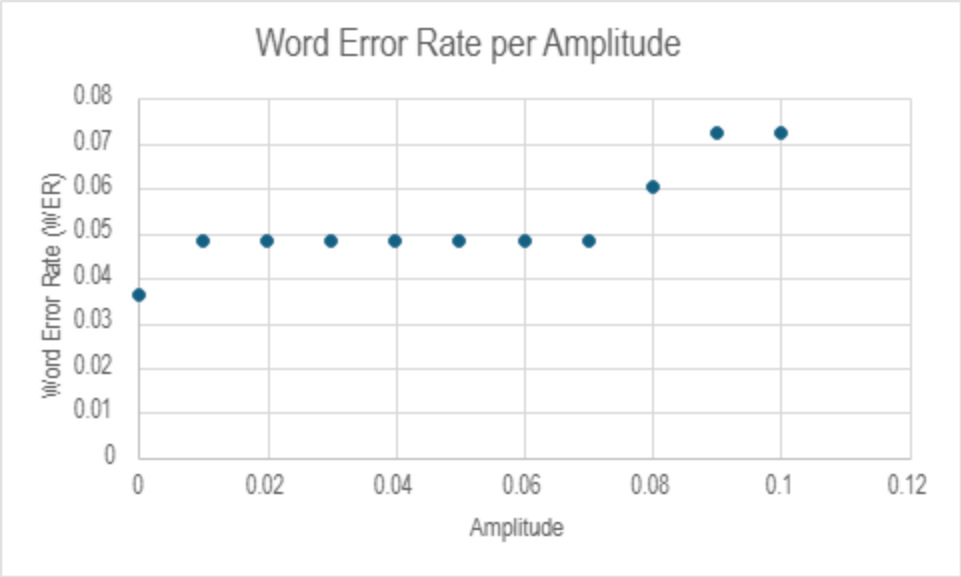
Expected Outcomes. The overall outcome of this aim is to allow for website interactions. This knowledge will be used to know the accuracy of the transcription models.

Potential Pitfalls and Alternative Strategies. We expect most of these cases to be correct as Selenium is powerful in detecting web page elements. We also expect most of the errors to be from transcription rather than Selenium performing the task.

Summary of Preliminary Data. Graph 1 shows the proportion of tested voices that were accurately detected under various background noises. It was made using audios of voices all saying the same sentence and then were marked to be correct or incorrect At low background noise levels, the recognition rate was nearly perfect. At higher background noise levels, the accuracy rate decreased. Graph 2 shows WER through varying levels of white noise, it shows a similar pattern of increasing errors through higher noise levels. However, the errors are low and show that the model can still be usable.



Graph 1: The accuracy rate of the speech recognition model under different background noise levels. This shows a decaying rate of accuracy but still being usable under average noise levels.



Graph 2: The Word Error Rate of the speech recognition model. Higher white noise levels increase the Error Rate of the speech recognition model.

Specific Aim #2: Cross Platform

Large assistants such as Siri and Google Assistant are not cross-platform. This can take away from the user experience as the latest features may not be available to them. The design of the program will be so that the features work in Windows and Mac operating systems. This wasn't an issue when it came to developing the website intractability portion of the app. However, it became tricky when it came to file management as different Operating Systems have different settings.

Justification and Feasibility. As said before, most of the features are incorporated well. However, managing files tends to be different. The os module in Python provided great features to the app for Windows. However, using the module sys made more sense as it has more features for cross-platform applications. Testing has only been done on a Windows 11 computer.

Potential Pitfalls and Alternative Strategies. The module os was difficult to use for cross-platform integration compared to sys. Therefore, a switch has been made. The current feature that allows users to open apps uses a module named subprocess and the sys module in order to perform the task. The sys module tells the subprocess the path to the application based on their operating system and then the subprocess module opens it.

Specific Aim #3: File Submissions

Websites such as Google Classroom and Canvas can require file submissions for assignments. Similarly, a lot of applications that can be used for work or school activities follow the same format. This is an issue that will use both sys and Selenium in order to complete given tasks. First, Selenium can select any button that allows someone to upload something, and then sys can locate files the user wants to upload, and modules such as PyAutoGui can be used to upload.

Justification and Feasibility. Managing local file systems has been accomplished in previous systems (Jain et al., 2021; Singh et al., 2022). This was done using the same tools I will, which proves that this task is possible.

Potential Pitfalls and Alternative Strategies. The only pitfall might be early integration into non-Windows operating systems due to their organization. However, these bugs will be resolved after further testing.

Section III: Resources/Equipment

All tests were performed on a Lenovo Slim Pro 7 laptop. The software used to program this were the Python 3.12 modules:

- Selenium
- Speech_recognition

- Sys
- Subprocess
- PyAutoGui
- TKinter

Visual Studio Code was used as an IDE for this project.

Section V: Ethical Considerations

Privacy issues are the only problems that may arise from this program. No voice recordings are being stored anywhere in the application. The program may have local file management features in the future. However, that can lead to a cyber security risk as hackers and people with malicious intent could try hacking into the application in order to reveal sensitive information about the user. This can be fixed by putting restrictions on the assistant such as a pin that would need to be communicated verbally in order to make sure that only the user is the only one using the file management features.

Section VI: Timeline

This research of other models started in September and ended in November. At the same time, the program was developed throughout the research projects, taking some inspiration from other projects. Data collection started in December and ended in February.

Section VII: Appendix

Criteria	Level	Google	Assembly	Whisper	Amazon Transcribe
Free	10	100	30	100	5
Accuracy	8	64	80	80	10
Latency	7	10	10	9	10
Noise Reduction	5	30	50	40	10

Language Support	3	24	7	10	10
Total		288	251	313	280

Table 1: Decision Matrix to justify using Google as a speech recognition model. Although it is the second-best pick in the decision matrix, it has a fast response in between inputs and outputs. Hence, it was used as a speech recognition model for this project.

Section VIII: References

Appalaraju, V., Rajesh, V., Saikumar, K., Sabitha , P., & Kiran, K. R.(2021, March 9). *Design and Development of Intelligent Voice Personal Assistant using Python*. 2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N), Greater Noida, India. pp. 1650-1654. IEEE. https://doi.org/10.1109/ICAC3N53548.2021.9725753.

Bajpai, D., Kiran, M. U., Reddy, B. H., & Natarajan, S. K. (2024, June 21-23). *Smart AI Voice Assistant through Generative Text Transformer and NLP Implementation in Python*. 2024 4th International Conference on Intelligent Technologies (CONIT), Bangalore, India, pp. 1-6. IEEE. https://doi.org/10.1109/CONIT61985.2024.10626557.

Barton, F. W. (2015). *Voice commands for transitioning between device states* (U.S. Patent No. 9,047,857B1). U.S. Patent and Trademark Office. https://patentimages.storage.googleapis.com/6d/9f/b5/6207989ec4793f/US9047857.pdf.

Chiusano, F. (2022, February 3). *Two minutes NLP — Intro to Word Error Rate (WER) for Speech-to-Text*. NLPlanet. https://medium.com/nlplanet/two-minutes-nlp-intro-to-word-error-rate-wer-for-speech-to-text-fc17a98003ea

Dilmegani, C. (2024, September 23). *Top 4 Speech Recognition Challenges and Solutions*. Research.aimultiple.com. https://research.aimultiple.com/speech-recognition-challenges/

Jain, R., Sharma, V., Mangilal, Kardam, R., & Rani, M. (2021). Artificial Intelligence Based A Communicative Virtual Voice Assistant Using Python & Visual Code Technology. *World Journal of Research and Review (WJRR), 13*(5), 23-25. https://www.wjrr.org/download_data/WJRR1305017.pdf.

Kunekar, P., Deshmukh, A., Gajalwad, S., Bichare, A., Gunjal, K., & Hingade, S. (2023, January 20-21). *AI-based Desktop Voice Assistant*. 2023 5th Biennial International Conference on Nascent Technologies in Engineering (ICNTE), Navi Mumbai, India. pp. 1-4. IEEE. https://doi.org/10.1109/ICNTE56631.2023.10146699.

Natural Readers. (2019). *text to speech online*. Naturalreaders.com. https://www.naturalreaders.com/

Prodeus, A., & Kukharicheva, K. (2016, October 18-20). *Training of automatic speech recognition system on noised speech*. 2016 4th International Conference on Methods and Systems of Navigation and Motion Control (MSNMC), Kyiv, Ukraine, pp. 283-286. IEEE. https://doi.org/10.1109/MSNMC.2016.7783147.

Selenium. (n.d.). *WebDriver*. Selenium. https://www.selenium.dev/documentation/webdriver/

Singh, S., Arora, D. K., Dar, I. N., Moghni, A., Kumar, S., & Kumar, A. (2022, February 23-25). *ARIA The Bot*. *2022 2nd International Conference on Innovative Practices in Technology and Management (ICIPTM)*, Gautam Buddha Nagar, India, pp. 167-174. IEEE. https://doi.org/10.1109/ICIPTM54933.2022.9753961.

Subhash, S., Srivatsa, P. N., Siddesh, S., Ullas, A., & Santhosh, B. (2020, July 27-28). *Artificial Intelligence-based Voice Assistant*. 2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4), London, United Kingdom. pp. 593-596. IEEE. https://doi.org/10.1109/WorldS450073.2020.9210344

Tinao, P., & Jamisola, R. S. (2023). Wildlife conservation using drones and artificial intelligence in Africa. *Science Robotics, 8*(85). https://doi.org/10.1126/scirobotics.adm7008.

Yip, M., Salcudean, S. E., Goldberg, K., Althoefer, K., Menciassi, A., Opfermann, J. D., Krieger, A.,

Swaminathan, K., Walsh, C. J., He (Helen) Huang, & Lee, I-Chieh. (2023). Artificial intelligence meets medical robotics. *Science*, *381*(6654), 141–146. https://doi.org/10.1126/science.adj3312