Screen Reading Enabled by Large Language Models

Large language models (LLMs), such as the pioneering GPT technology by OpenAI, have undeniably become one of the most significant innovations in recent history. They have achieved phenomenal success across a broad spectrum of applications in numerous industries, transforming how we interact with the digital world. Notwithstanding these remarkable successes, the application of LLMs within the realm of accessibility has largely been unexplored. We introduce Savant, as a demonstration of the potential of LLMs for accessibility. Specifically, Savant leverages the impressive text comprehension abilities of LLMs to provide uniform interaction for screen reader users across various applications, mitigating the significant interaction burden imposed by the heterogeneity in user interfaces for blind screen reader users. Savant automates screen reader actions on control elements like buttons, text fields, and drop-down menus via spoken natural language commands (NLCs). Interpreting the NLC, identifying the correct control element, and formulating the action sequence are facilitated by LLMs. Few-shot prompts supply context and guidance for the LLMs to produce appropriate responses, specifically converting the NLC into a correct series of actions on the user interface elements, which are then performed automatically. The demonstration will exhibit Savant's capability across a variety of exemplar applications, emphasizing its versatility.

Additional Key Words and Phrases: Blind users, Computer Interaction, Accessibility, Assistive technology, Uniform interaction, Large language models (LLMs)

ACM Reference Format:

. 2024. Screen Reading Enabled by Large Language Models. 1, 1 (July 2024), 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Proficiency in computer software has become indispensable. To maintain competitiveness within the educational and professional domains, the acquisition of computer skills is paramount. Indeed, numerous institutions of higher education and professional occupations necessitate a comprehensive understanding of computer software, such as the Office suite and other well-established programs [2, 8].

For blind users, interacting with these applications can be difficult and time-consuming because their graphical user interfaces (GUIs) are primarily designed for visual interaction, typically in the form of a two-dimensional GUI that interacts with a mouse and keyboard. Blind users depend on assistive technology, specifically a screen reader JAWS [1], VoiceOver [2], and NVDA [3]), to engage with computer applications. The screen reader reads the content on the screen out loud, allowing blind users to navigate and interact with different control elements and content sections of the application using keyboard shortcuts. However, most modern computer applications primarily feature tightly clustered GUIs intended for pointer interactions using a mouse, posing challenges for blind individuals who rely heavily on keyboard shortcuts for interaction.

[1] https://www.freedomscientific.com/products/software/jaws/
[2] https://www.apple.com/accessibility/mac/vision/
[3] https://www.nvaccess.org/

Author's Contact Information:

Manuscript submitted to ACM

The heterogeneity of application interfaces, each necessitating the memorization of an array of keyboard shortcuts for analogous tasks, introduces an additional layer of complexity to the interaction challenges confronted by blind users. For instance, to perform a simple task like opening an incognito window, the keyboard shortcut used in *Google Chrome* is *Ctrl+Shift+N* whereas for *Mozilla Firefox*, it is *Ctrl+Shift+P*. A blind user must be able to recall these shortcuts correctly, as pressing *Ctrl+Shift+P* in *Google Chrome* would open the printing dialog box instead. Such cases might interfere with the user's task and reduce their overall productivity. Furthermore, different operating systems often have different keyboard shortcuts to perform the same task in the same application, and these shortcuts also vary significantly between different screen readers. MacOS often employs the *Command* and *Option* keys in keyboard combinations for shortcuts, whereas Windows users must use keys such as the *Ctrl* and *Alt* keys in their combinations.

Blind participants in a previous study highlighted the main limitations they found in the assistive technology available today and the importance of having consistent and standardized access across different platforms, screen readers, and applications [3]. Blind users desire consistent access, yet the methodologies to offer it are still relatively undeveloped. However, the recent emergence of Large Language Models (LLMs), such as GPT-4 [1], Gemini [6], and Llama-2 [7] offers a very promising path to address the heterogeneity problem.

We introduce Savant, as a demonstration of the potential of LLMs for accessibility. Specifically, Savant leverages the impressive text comprehension abilities of LLMs to provide uniform interaction for screen reader users across various applications, mitigating the significant interaction burden imposed on them by the heterogeneity in user interfaces. Savant automates screen reader actions on control elements like buttons, text fields, and drop-down menus via spoken natural language commands (NLCs).

Interpreting the NLC, identifying the correct control element, and formulating the action sequence are facilitated by LLMs. Few-shot prompts [4] supply context and guidance for the LLMs to produce appropriate responses, specifically converting the NLC into a correct series of actions on the user interface elements, which are then performed automatically. The demonstration will exhibit Savant's capability across a variety of exemplar applications, emphasizing its versatility.

2 Technical Design of Savant

Figure 1 depicts an architectural schematic of Savant. The user can activate it using a special keyboard shortcut "Alt+Insert".

Savant consists of two major components - (1) Pre-processing component and (2) Run-Time component.

The tasks of the Pre-processing component include: (1) Creating a non-visual hierarchical representation of the application GUI (depicted as "Application Control Tree" in the figure). This contains all the information required to generate a series of steps on the UI to trigger an action on a control element. (2) Creating a dataset of few-shot examples for the application to serve as prompts (referred to as "Few-Shot Example Database" in the figure).

The *Crawler* module constructs the Application Control Tree (ACT) by leveraging UI Automation Tools [5]to extract all information regarding the control elements in the application, including metadata (e.g., control element, actions on the control elements like clicking, selection, hierarchical relationships between control elements, etc.) The *Pair-Generator* module uses the metadata of the ACT to generate <control element (α), value (val)> pairs for each control element α in the application and saves it.We associate an Action with every pair <α, val>, meaning that execution of the Action will result in a sequence of screen reader steps that will result in the assignment of the val to α.

For every <α, val> pair in the application, the *Seed LLM* generates few-shot example. This is then stored as a <NLC, α, val> triplet in the Few-shot Example Dataset (FED). We provide prompts to the *Seed LLM*, which includes <α, val> pairs from the pair generator and hand-crafted examples. The hand-crafted few-shot examples guide the *Seed LLM* in

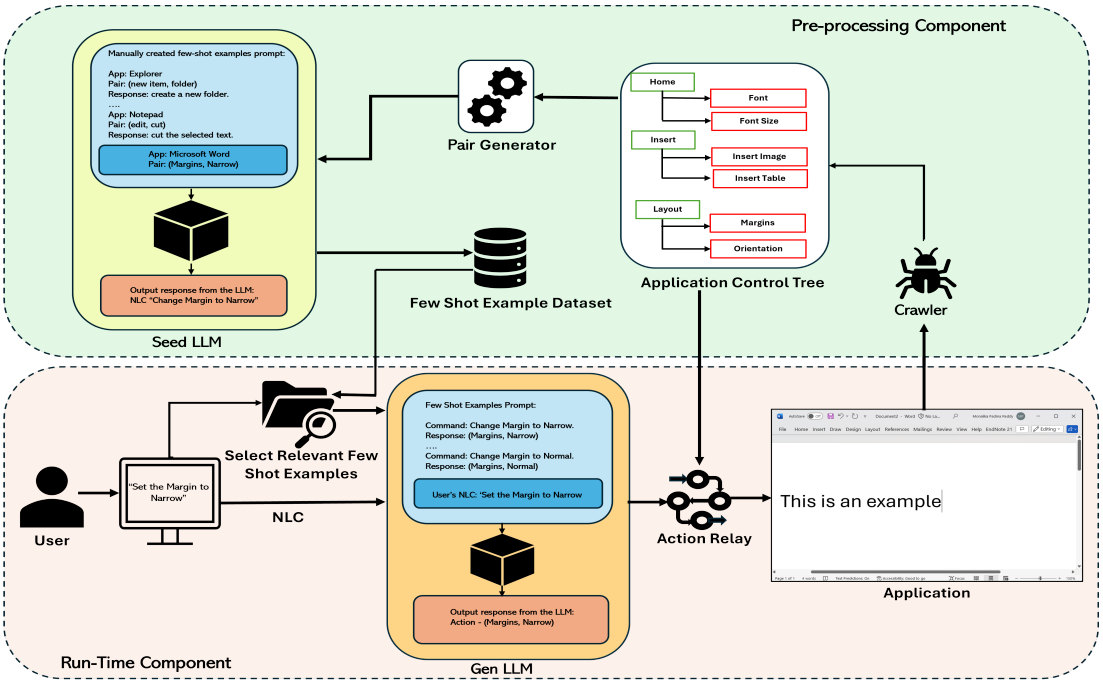

Fig. 1. Savant System Design and Architectural Workflow.

generating the NLCs for each <α, val> pair. The Pre-processing for each application is done *only once*. During runtime, the Application Control Tree (ACT) and Few-Shot Examples Dataset (FED) are used repeatedly to generate the sequence of screen reader steps needed to execute the user's NLC.

The Run-Time component interprets the NLC given by the user, generates corresponding <α, val>, for the NLC, associate an Action on the generated <α, val>, pair and executes a sequence of screen reader steps for the corresponding action. The Gen LLM interprets and maps the NLC to the appropriate <α, val> pair. The prompt to the Gen LLM comprises the user's NLC and a set of few-shot examples relevant to the user's NLC. The Action Relay generates the sequence of steps for the screen reader to execute the action on the <α, val> pair, using the ACT.

3 Demonstration Scenarios

Through the scenarios outlined below, we showcase how Savant assists screen reader users by allowing them to uniformly and effectively navigate any application's interface with natural language commands. It eliminates the need for users to remember different shortcuts for various applications. Savant's ability to interpret NLCs without requiring users to remember the precise names of control elements underscores the power and flexibility of leveraging LLM. Moreover, Savant demonstrates advanced semantic understanding, allowing it to accurately process different expressions of the same command. This functionality enables users to engage with applications in a manner that is both natural and intuitive, thereby enhancing their interaction experience.

3.1 Scenario 1

Bob, a blind individual, is creating a report using the *MS Word* application. He wants to insert a table with three rows and five columns as seen in Figure 2.

If Bob solely relied on screen readers to perform this task, he would first use *Alt* key to activate the Menu Ribbon and press *N* to open the *Insert* tab. Next, he would use the arrow keys to navigate to the *Table* option, then press *Enter* to select it. Subsequently, he would use the arrow keys to choose 3 rows and 5 columns in the grid and then finalize the selection by pressing Enter once more.

Bob completes the same task on by saying "Create a new table with 3 rows and 5 columns". Savant interprets his command, identifies his intent, and translates it into a specific action—inserting a 3x5 table. i.e (table,3x5 table). It identifies the series of screen reader steps required to insert the table, i.e., *Select the menu ribbon, Select the insert tab, Select Table, and Select* 3 *rows and* 5 *columns from the dropdown.* Even if Bob says "Insert a 3 x 5 table" or "Insert the table with 3 rows and 5 columns", Savant would still interpret the commands accurately and generate the same sequence of screen reader actions on the Word application interface.

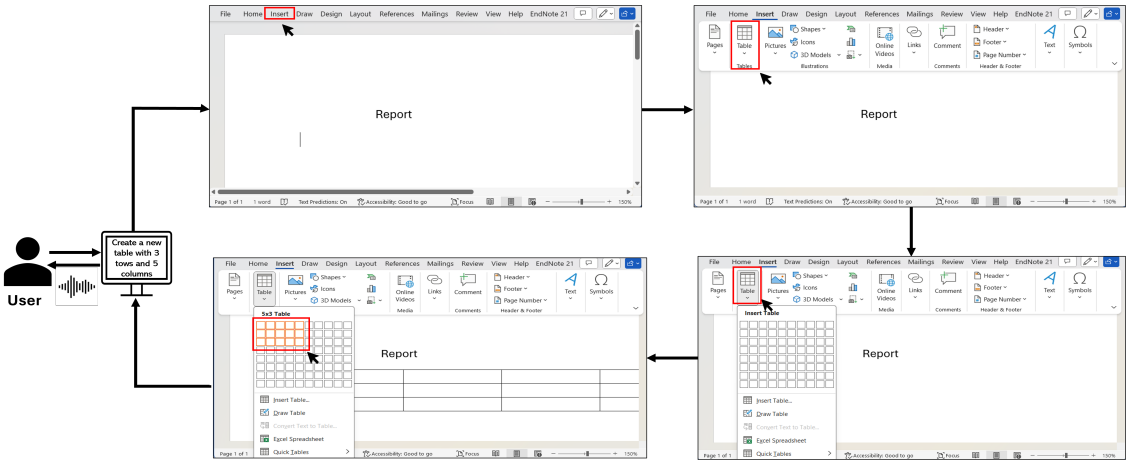

Fig. 2. Demonstration on *MS Word* to insert a table

3.2 Scenario 2

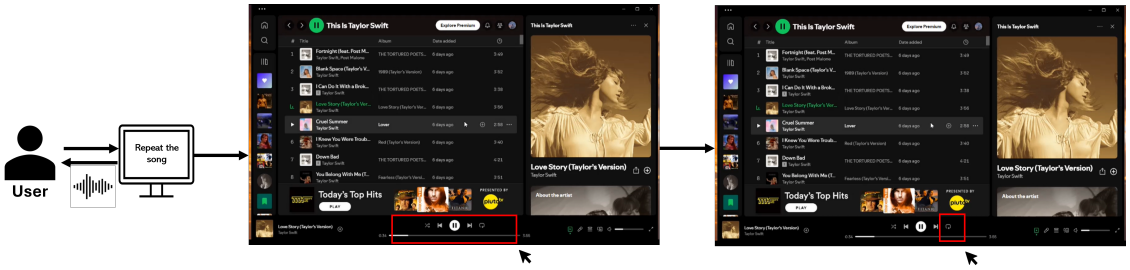

Fig. 3. Demonstration on Spotify to repeat the current song.

Claire, who is also blind, is sitting in her home office listening to music on Spotify, practicing for an upcoming audition. As the first song nears its end, she decides to replay it and focus more on the high notes.

If Claire were to use a screen reader to replay the song, she would have to navigate to the “Enable repeat button” using the Tab key and hit enter. Instead, she simply commands Savant to “Repeat this song”. Savant identifies the intent of her command, which is to “Repeat the song that is currently playing”, and converts it into an action on the control element using the control-value pair <*enable repeat*,*none*>. Then, it simulates a click on the “Enable repeat button”.

3.3 **Scenario** 3

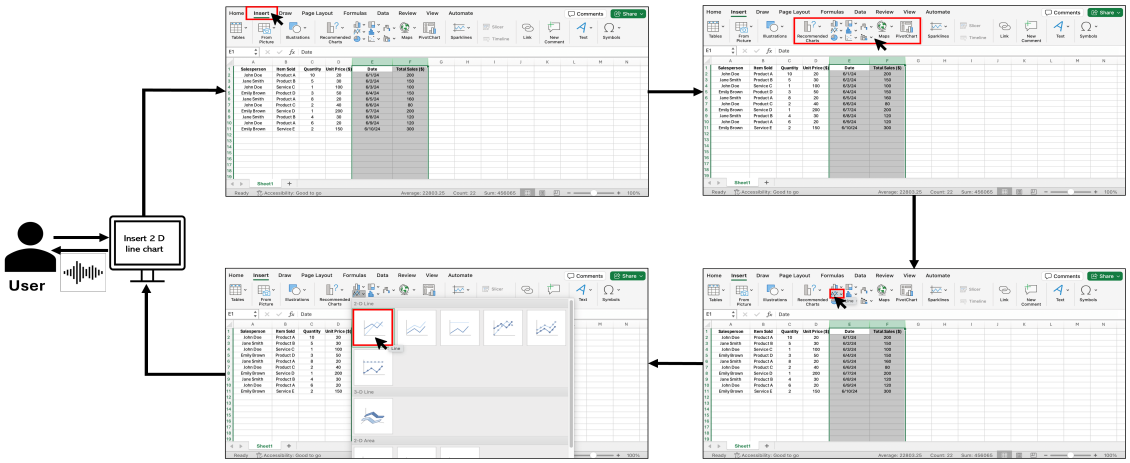

Fig. 4. Demonstration on Excel to insert a Line Chart using the selected data columns.

Daphne, a visually impaired individual is preparing a monthly sales report on *MS Excel*. She will present it to her team at the weekly sales meeting. She wants to include a line graph illustrating the sales in the previous month.

Should Daphne decide to rely on a screen reader to assist her in inserting the line graph, she would first choose the columns “Date” and “Total Sales”. Then, she would activate the Menu ribbon and select the Insert Tab using the keyboard shortcut *Alt+N*. Subsequently, she would use the arrow keys to navigate to the line chart, choose the 2 D Line chart, and hit Enter.

Alternatively, using Savant, as seen in Figure 4 Daphne simply needs to highlight the columns labeled “Date” and “Total Sales” and say “Insert a 2 D Line chart”. In response, Savant generates the control value pair <*insert line or area chart*, *line*>. It identifies and sequentially executes the following sequence of screen reader steps: *Select the Menu ribbon, click Insert tab, click Line or Area chart, Select 2 D Line chart.*

4 Evaluation

We conducted an IRB-approved user study with 11 visually impaired screen reader users to evaluate our system’s effectiveness. The results revealed that the performance of Savant was 76.7% faster compared to JAWS, and its task completion rate was 81%. The Keypress statistics showed that *AccessBolt* [8] (70%) and JAWS,(91%) yielded higher percentages;. At the same time Savant’s (30%) decreased reliance on navigation keys like the Tab and Arrow Keys indicates improved effectiveness and increased efficiency. The participants gave positive feedback, which included appreciating the system’s user-friendliness and fewer keystrokes. Additionally, they found the voice interface beneficial.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).

[2] Edward C Bell and Natalia M Mino. 2015. Employment outcomes for blind and visually impaired adults. (2015).

[3] Syed Masum Billah, Vikas Ashok, Donald E Porter, and IV Ramakrishnan. 2017. Ubiquitous accessibility for people with visual impairments: Are we there yet?. In *Proceedings of the 2017 chi conference on human factors in computing systems*. 5862–5868.

[4] Ben Mann, N Ryder, M Subbiah, J Kaplan, P Dhariwal, A Neelakantan, P Shyam, G Sastry, A Askell, S Agarwal, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165* 1 (2020).

[5] Microsoft. 2021. *Microsoft UI Automation.* https://docs.microsoft.com/en-us/windows/win32/winauto/entry-uiauto-win32 Accessed: 2024-07-02.

[6] Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805* (2023).

[7] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* (2023).

[8] Utku Uckun, Rohan Tumkur Suresh, Md Javedul Ferdous, Xiaojun Bi, IV Ramakrishnan, and Vikas Ashok. 2022. Taming User-Interface Heterogeneity with Uniform Overlays for Blind Users. In *Proceedings of the 30th ACM Conference on User Modeling, Adaptation and Personalization*. 212–222.