VoiceAlign: A Shimming Layer for Enhancing the Usability of Legacy Voice User Interface Systems

Md Ehtesham-Ul-Haque
Pennsylvania State University
University Park, Pennsylvania, USA
mfe5232@psu.edu

Syed Masum Billah
Pennsylvania State University
University Park, Pennsylvania, USA
sbillah@psu.edu

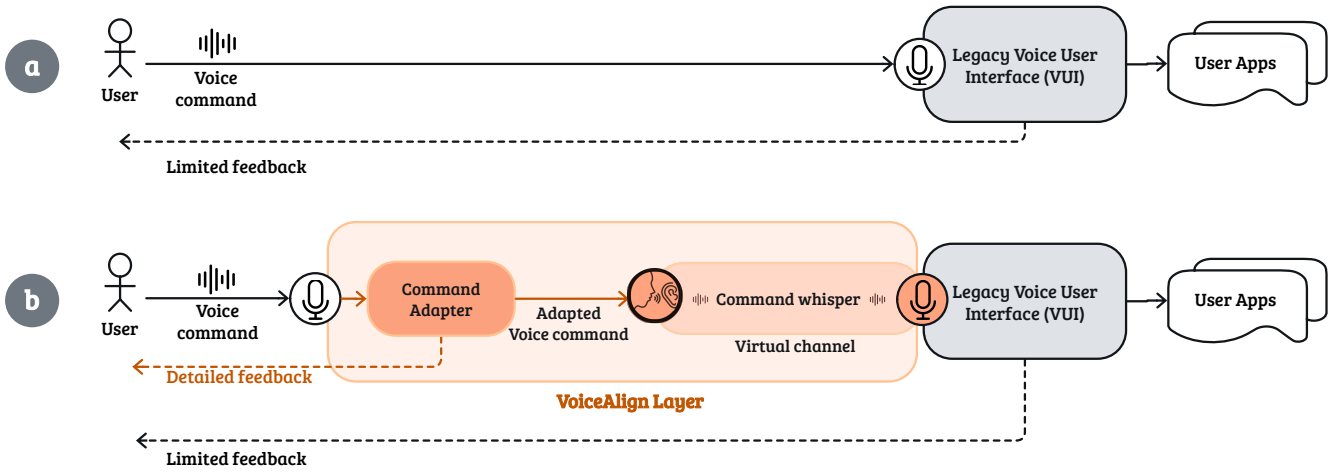


Figure 1: An illustration of VoiceAlign's integration with legacy voice user interface (VUI) systems. (a) In a conventional legacy VUI system, users must utter fixed-syntax commands to interact with applications. The VUI interprets these commands and executes them only when they match exact syntactic and semantic requirements, discarding imperfect attempts. (b) VoiceAlign (in orange) functions as a shim layer between system audio input and the legacy VUI. It intercepts user commands, adapts them to match the required syntax and semantics, and relays these corrected commands through a virtual audio channel. This approach frees users from memorizing exact command phrasing and eliminates the need to restate commands during multi-step tasks.

Abstract

Voice user interfaces (VUIs) are rapidly transitioning from accessibility features to mainstream interaction modalities. Yet most operating systems' built-in voice commands remain underutilized despite possessing robust technical capabilities. Through our analysis of four commercial VUI systems and a formative study with 16 participants, we found that fixed command formats require exact phrasing, restrictive timeout mechanisms discard input during planning pauses, and insufficient feedback hampers multi-step interactions. To address these challenges, we developed VoiceAlign, an adaptive shimming layer that mediates between users and legacy VUI systems. VoiceAlign intercepts natural voice commands, transforms them to match the required syntax using a large language model, and transmits these adapted commands through a virtual audio channel that remains transparent to the underlying system. In our evaluation with 12 participants, VoiceAlign reduced command failures by half, required 25% fewer commands per task, and significantly lowered cognitive and temporal demands when paired with an existing legacy VUI system. Furthermore, we created a synthetic dataset informed by our studies and fine-tuned a small language model that achieves over 90% accuracy with 200 ms response time when served locally, eliminating dependence on third-party APIs while enabling real-time interaction on edge devices. This work demonstrates how modern AI techniques can unlock the underutilized potential of legacy VUI systems without requiring system modifications, offering a practical solution without replacing existing infrastructure.

CCS Concepts

• **Human-centered computing** → **Interaction techniques**; *User centered design.*

Keywords

Voice commands, voice user interface, dictation, text correction, shim layer.

ACM Reference Format:
Md Ehtesham-Ul-Haque and Syed Masum Billah. 2026. VoiceAlign: A Shimming Layer for Enhancing the Usability of Legacy Voice User Interface Systems. In *31st International Conference on Intelligent User Interfaces (IUI*

IUI '26, Paphos, Cyprus

ACM ISBN 979-8-4007-1984-4/2026/03
https://doi.org/10.1145/3742413.3789167

'26), March 23–26, 2026, Paphos, Cyprus. ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3742413.3789167

1 Introduction

Voice user interfaces (VUIs) have long served as accessible alternatives to pointing- or touch-based direct manipulation of graphical interfaces [58]. Historically, users with blindness or upper-limb disabilities relied on VUIs more frequently than their sighted, non-disabled peers [9, 55]. However, voice input is rapidly shifting from an accessibility feature to a mainstream interaction modality. This transition stems from several converging factors: voice input becoming ubiquitous across devices from smart homes to smart glasses, emerging use cases such as coding where typing complex syntax proves cumbersome, and latency improvements enabling real-time interaction. While VUIs generally prove less efficient than direct manipulation for traditional interaction-oriented activities [37], they offer unique benefits for hands-free control across diverse environments, including healthcare facilities and augmented reality systems.

This mainstreaming of voice input creates an urgent need to address a persistent paradox: most operating systems include built-in voice command systems that remain underutilized despite their robust technical capabilities. VUI systems like Voice Control [3], which enables speech-based interaction as an accessibility feature across Apple's macOS, iOS, and visionOS platforms, possess powerful functionality, including direct access to accessibility APIs, DOM tree structures, and system-level control. Yet these systems require fixed command formats with precise syntax and terminology, creating a fundamental mismatch with natural speech production, which is more verbose, contains disfluencies and false starts, and follows looser organizational structures [23, 29]. Unlike text-based disambiguation, where users can pause indefinitely to review dropdown menus or select from clickable candidate lists, voice commands are ephemeral and temporally constrained — users cannot visually inspect alternatives before committing, timeout windows prevent leisurely selection processes, and acoustic uncertainty from ASR errors compounds the challenge [23, 48]. This fundamental difference imposes cognitive burdens as users must recall exact phrasings in real-time, while minor command variations lead to immediate failures and retries. While modern VUI platforms like Alexa [45] and ChatGPT desktop [36] have advanced toward natural language understanding [31], organizations hesitate to replace legacy VUI systems due to prohibitive costs and operational disruptions [53]. These legacy systems function reliably when users conform to their preconditions, but mastering those preconditions — speaking in ways contrary to natural speech patterns — challenges end users [18]. A wrapper approach, therefore, offers a more practical solution than complete replacement, as it preserves existing systems while enhancing usability.

To inform our design, we explored the voice command structures across four commercial VUI systems and performed an in-depth analysis of text correction commands supported by all four systems. We then conducted a formative study with 16 participants performing common text correction tasks. Our command structure analysis revealed inconsistencies across command types for target acquisition, target manipulation (e.g., delete a word), and system control. Findings from our formative study showed that participants found the commands restrictive and programming-like, frequently producing minor command variations that VUIs could not support, hampering usability and increasing cognitive demand. Additionally, restrictive timeout mechanisms discarded partial input when users paused to plan multi-component commands, while limited system feedback was insufficient to guide users through error recovery or clarify command requirements. Text correction tasks amplified these challenges because when commands failed due to out-of-vocabulary words, timeouts, or ASR errors, they created error cascades requiring additional corrective commands, representing a worst-case scenario for evaluating VUI usability.

These findings led us to develop **VoiceAlign**, an adaptive shimming layer that mediates between users and legacy VUI systems. As Figure 1 illustrates, VoiceAlign consists of two components: a *command adapter* that intercepts the system microphone and uses an LLM to map users' utterances to the correct command syntax, and a *command whisperer* that relays the adapted commands to the legacy VUI through a virtual audio channel. The system provides real-time visual feedback throughout command utterance and employs extended timeout windows to accommodate planning time.

A summative evaluation with 12 participants demonstrated that VoiceAlign reduced command failures by half through suggestions and real-time feedback, and required 25% fewer commands per task while significantly lowering cognitive load when mediating between the user and a legacy VUI. To enable practical deployment and improve response time, we fine-tuned a 270M parameter language model that achieves over 90% accuracy in command adaptation with a 200 ms response time when served locally, eliminating dependence on external APIs while enabling real-time interaction on edge devices.

In short, our contributions are:

- Analyzed command structures of four legacy VUI systems for consistency patterns and conducted a formative study with 16 participants, identifying command utterance strategies and expectation-structure mismatches for text correction tasks.
- Proposed VoiceAlign, a shim layer architecture enhancing legacy VUI usability through command adaptation and virtual audio channeling.
- Evaluated VoiceAlign with 12 participants, demonstrating improved usability without requiring complete system replacement.
- Generated a synthetic dataset for command correction informed by study data and fine-tuned a small model capable of running locally and responding in real-time with high accuracy.

2 Background and Related Work

In this section, we describe prior work in voice-/speech-based text correction systems, the correction instructions of the commercially available systems and their challenges, multi-modal correction systems to improve voice commands, and unique challenges in disambiguating voice input.

2.1 Voice-based Text Correction

Text correction usually requires users to specify two pieces of information. First, they need to specify the target location where the correction will occur (target acquisition). In addition, they need to convey what changes will be performed to correct the text (correction). In voice-based correction systems, users can acquire the target location through instructions that move the cursor to the target location (navigation-based) or select the target phrase (selection-based). For correction, users must specify the correction type through a command (e.g., delete, insert, or replace) and additional arguments (e.g., the phrase users want to insert) if required.

Speech-based systems use automated speech recognition (ASR) to capture these dictated commands and arguments. Two speech-based techniques are commonly used for correction - command-based and re-dictation [16]. Command-based techniques support a set of fixed-format correction instructions containing a command, a context (e.g., a location), and one or more arguments (e.g., `select <phrase>` and `insert <phrase> before <phrase>`). These systems utilize a parser to detect valid commands and the arguments to manipulate the text [16]. Commands are used for both target acquisition and correction. Corrections can be performed in two steps - using a selection instruction to acquire the target (`select <phrase>`) followed by a correction instruction (`delete that`). These two steps can also be combined into a single step (`delete <phrase>`). Re-dictation replaces the target with the dictated phrase without any predefined commands [16]. This correction type can also work in either two steps or one step. Two-step re-dictation also requires target acquisition through a command followed by dictating the correct phrase [18]. In one-step re-dictation, users only dictate the correct phrase, and an underlying algorithm decides the target that aligns the most with the dictation [33]. This is also known as fluid [60]/seamless [6] correction or correction through re-speaking [54, 59]. Ghosh et al. [14] found that command-based techniques allow more control to users and are preferable for correcting single-word errors, whereas re-dictation approaches are more natural and effective in correcting multi-word errors.

2.2 Usability Issues of Voice Commands for Text Correction

Prior work has investigated the challenges of speech-based text correction. Although speech is faster than keyboard and mouse, correcting errors with dictation is significantly slower [18]. There are two sources of errors in dictation systems. The first is direct/user errors, such as invalid commands or wrong words due to stuttering or pausing. The other is indirect/system errors when the ASR fails to detect valid instructions or misrecognizes them due to difficulty separating commands from dictation, noise, or non-native pronunciation [8, 14, 18, 23, 44, 48].

Unrecognized or misrecognized voice commands can significantly limit the usability of speech-based text correction. Unrecognized commands confuse users, leading to repetitions (spiral depth) [18, 44, 48] while misrecognition can introduce new errors (cascading effect) [8, 18, 42], making recovery difficult. Sears et al. [48] found navigation commands misrecognized as dictation, requiring undo operations, and users spent 66% of their time correcting errors.

Most prior research analyzing voice commands was conducted in the early days of developing ASR and dictation systems. They investigated dictation systems with limited features (e.g., using only the `correct` command for making corrections [18]) or focused on a limited scope (e.g., investigating only navigation instructions [48]). In addition, there is a dearth of research investigating the current speech-based text correction systems built on state-of-the-art ASR systems that include many correction commands. This work aims to fill this gap by comprehensively studying commercial speech-based dictation systems and investigating their usability and challenges.

2.3 Improving Voice-based Text Correction with Additional Modality

Although speech is convenient, faster, and more intuitive, it is prone to recognition errors. In addition, proactive control of the microphone is inconvenient and can lead to unintentional command invocation. To mitigate this problem, prior work explored complementing voice commands with additional modalities (e.g., gaze, touch, and gesture), which can lead to effective and robust interaction [68]. Multimodal interaction can support natural [24, 67] and robust experience [39, 40], provide flexibility [11, 41], and can reduce cognitive load [57]. Consequently, researchers explored multimodal systems in various text-based interaction tasks, such as using pen-based gestures to correct ASR errors [56], combining gesture and speech for typing [25, 51], using touch to indicate speaking word boundaries for improving ASR performance [38, 50], and combining eye gaze and a desktop keyboard for text editing [52].

Multimodal text correction adopts a two-step dictation-based process, where target acquisition is separated from speech and attributed to a suitable modality (touch, eye gaze, and gesture). Most of the prior work leveraged touch as the modality for target acquisition. For example, Zhao et al. [66] improved smartphone text correction by using touch for easier target acquisition and speech for correction operations. Touch and speech have also been found to be practical for text modification inside hypothetical automated vehicles. TouchEditor [64] used touch gestures (swipe and shape) on a wearable piezoresistive sensor to correct texts on head-mounted devices in speech-unfriendly environments. Ghosh et al. [15] explored text editing in smart glasses while walking using speech and a hand-held remote.

Researchers have also explored other modalities in mid-air interaction where touch or a keyboard is not available. For example, Talk-and-Gaze [49] and EyeSayCorrect [67] leveraged eye gaze for target acquisition and speech dictation. For post-editing machine translation, bimanual gestures and speech have also been explored in prior work using gesture elicitation studies [20, 22].

While these multimodal systems can make speech-based interaction robust and improve usability, they may not be usable for users who cannot interact with that modality due to impairments. Consequently, making speech-only instructions robust and usable is essential.

2.4 Challenges of Disambiguating Voice Input

While disambiguation interfaces have been extensively studied in text-based contexts—including dropdown refinement [19], query suggestions [2], and candidate selection interfaces—voice command

correction presents fundamentally distinct challenges. Unlike text input, where users can pause indefinitely, visually inspect their typed query, and iteratively refine character-by-character, voice commands are ephemeral and temporally constrained — users must articulate complete multi-component commands within strict timeout windows (typically 1-2 seconds between components) or lose all progress [48]. This temporal pressure is compounded by acoustic uncertainty, as ASR errors, homophones, and pronunciation variations introduce ambiguity before semantic interpretation begins—a layer of uncertainty absent in text interfaces where input fidelity is guaranteed [23].

Moreover, the spontaneous nature of speech production itself complicates disambiguation. Unlike text, where users can draft, review, and edit before submission, voice commands emerge in real-time with disfluencies, false starts, and self-corrections that are natural artifacts of unplanned speech [7, 29]. Users cannot reliably produce perfectly formed commands without advance planning. This challenge is further exacerbated by the difficulty of maintaining command mode boundaries — users must continuously manage whether they are issuing commands or thinking aloud, as systems cannot reliably distinguish intentional commands from planning utterances, self-talk, or environmental noises [30, 40].

Our work addresses these voice-specific challenges through a shimming layer that preprocesses commands before passing them to the legacy VUIs. We provide extended timeout windows to accommodate spontaneous speech planning, use LLM-based parsing to separate noise and disfluencies from command components and map them to correct syntax, and generate clarifying questions for incomplete commands rather than discarding partial input.

3 Analyzing Commands in Legacy VUI Systems

To better understand VUI commands, their structures, and their applications in personal computer use, we conducted a systematic analysis of their command structures. We first identified VUI systems with fixed-format commands that met four criteria: (i) fully-featured with comprehensive command sets, (ii) widely available to users, (iii) commercially popular with substantial user bases, and (iv) supporting interaction across common operating systems. This selection process yielded four systems: Voice Control (Apple) [3], Dragon Speech Recognition (Nuance) [35], Voice Access for Windows (Microsoft) [34], and Voice Access for Android (Google) [17]. We then analyzed the command capabilities of all four systems and identified that dictation and text correction commands were universally supported. Therefore, we focused on text correction tasks with voice commands (Table 1), which prove especially challenging with voice [8, 18, 48].

Next, each author independently analyzed the commands supported by each system, using them to perform common text correction tasks (e.g., selecting, deleting, inserting, replacing, and correcting text) to understand in depth how each command functioned. During this analysis, the authors sought to identify command structures, their intricacies and components, what factors made commands successful or caused them to be discarded by the system, and the feedback provided during interaction. Each author took detailed notes independently and created command templates based on their analysis. Following individual analysis, the authors met to discuss their notes, identify common themes regarding text correction commands use, and consolidate their templates into a unified, fully-quantified command template that specifies valid component combinations. We discuss this template, its components, valid combinations, and command utterance techniques in detail in this section.

Fully Quantified Command Template. Legacy VUI systems require users to memorize and execute precise command structures with fixed syntax. These systems typically organize commands into distinct operational categories (Table 1): mode switching, phrase dictation, navigation, selection, and manipulation (insertion, deletion, replacement). We generalize their command structures into four components, as shown in Figure 2: `[command (cmd)] <command argument (cmd-arg)> [context (ctx)] <context argument (ctx-arg)>`.

Using this template, we identified six common combinations (Fig. 3) that use various subsets of these four components. While some command structures remain consistent across all four systems, significant variations exist, especially in complex operations. For example, replacement operations use different syntactic structures: These variations highlight the irregularity, implicit complexity, and ad hoc nature of command structures in legacy VUI systems.

Command Complexity. For any command to succeed, users must correctly populate all required components with valid information. We rank the complexity of commands based on their component combinations:

- *Single-component commands* like mode switching impose minimal cognitive load.
- *Simple commands* for selection, deletion, or correction typically require two components: a command keyword and a parameter.
- *Complex commands* for insertion or replacement require all four components (e.g., `insert <word> before <word>`), creating maximal cognitive load by requiring users to organize multiple pieces of information in the correct sequence.
- *Special cases* exist, such as the "`choose`" command, where the first component (command keyword) becomes optional, allowing users to select by simply speaking a number.

Timeout Mechanisms and Their Impact. A critical usability constraint in legacy VUI systems is the timeout mechanism. As shown in Figure 2, each transition between components has an associated timeout threshold (inter-component; 1 to 2 seconds). This timeout also applies to an individual component with multiple parts (intra-component; e.g., a phrase as a `cmd-arg`). If users pause longer than the allowed threshold, the system discards the partial command and often interprets subsequent speech as new input.

4 Formative User Study

We conducted an IRB-approved study with 16 participants (14 males, 2 females, ages 24-35) recruited through university mailing lists and word of mouth. All participants were fluent English speakers with varying accents, without speech difficulties, and had prior experience with voice interfaces such as Siri, Google Home, or Alexa. We refer to participants as P1 through P16 throughout our analysis.

Operation	Voice Control (Apple)	Dragon Speech (Nuance)	Voice Access for Windows (Microsoft)	Voice Access for Android (Google)
Mode Switching	• dictation mode • command mode • spelling mode	• switch to dictation mode • switch to command mode • switch to spelling mode	• default mode • commands mode • dictation mode	
Phrase Dictation	• <phrase> • type <phrase>	• <phrase>	• <phrase> • type <phrase> • dictate <phrase>	• <phrase> • type <phrase>
Navigation	• move before <phrase> • move after <phrase>	• move before <n> characters • move down <n> lines	• move before <phrase> • move after <phrase>	• move before <phrase> • move after <phrase> • move between <phrase> and <phrase>
Phrase Selection	• select <phrase> • select previous word • select next word	• select <phrase> • select <phrase> through <phrase> • select next <n> words	• select <phrase> • select previous word • select next word	• select <phrase> • select from <phrase> to <phrase>
Number Selection	• choose <number> • <number>	• choose <number> • <number>	• click <number> • <number>	
Insertion	• insert <phrase> before <phrase> • insert <phrase> after <phrase> • Cursor Movement → Phrase Dictation	• insert before <phrase> → Phrase Dictation • Cursor Movement → Phrase Dictation	• insert before <phrase> → Phrase Dictation • insert after <phrase> → Phrase Dictation • Cursor Movement → Phrase Dictation	• Insert <phrase> before <phrase> • insert <phrase> after <phrase> • insert <phrase> between <phrase> and <phrase>
Deletion	• delete <phrase> • Phrase Selection → delete selection • Phrase Selection → delete that	• delete <phrase> • delete from <phrase> to <phrase> • Cursor Movement → delete last <n> words • Cursor Movement → backspace <n>	• delete <phrase> • Phrase Selection → delete that • Phrase Selection → scratch that • Phrase Selection → strike that	• delete <phrase> • delete from <phrase> to <phrase> • Phrase Selection → delete selected text
Replacement	• replace <phrase> with <phrase> • change <phrase> to <phrase> • Phrase Selection → Phrase Dictation	Phrase Selection → Phrase Dictation	Phrase Selection → Phrase Dictation	• replace <phrase> with <phrase> • replace everything between <phrase> and <phrase> with <phrase>
Fixing	• correct <phrase> • Phrase Selection → correct that	• correct <phrase> • Phrase Selection → correct that	• correct <phrase> • Phrase Selection → correct that	

Table 1: A sampler of commands supported by four commercial VUI systems.

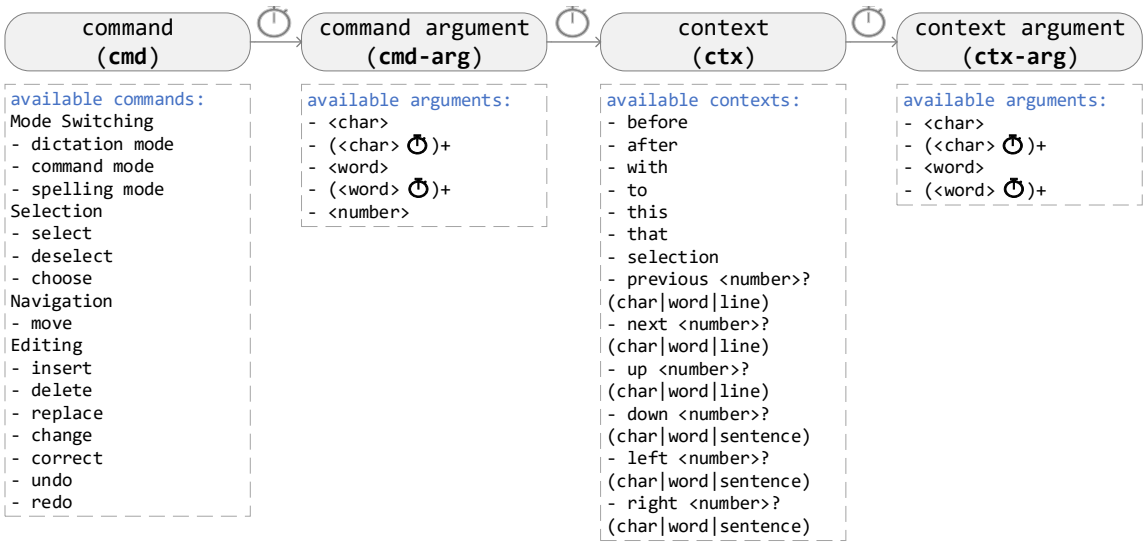


Figure 2: The fully quantified command template with four components and a sampler of available values for each component. An icon indicates the timer that specifies the threshold between two utterances to be considered part of the same instruction. Note that the timer applies to both inter- and intra-component utterances.

Combination	Example Commands
command (**cmd**)	- dictation mode - command mode - spelling mode
command argument (**cmd-arg**)	- <number>
command (**cmd**) + command argument (**cmd-arg**)	- select <word> - choose <number> - delete <word> - correct <word>
command (**cmd**) + context (**ctx**)	- select that - select previous word - select next word - delete that - delete next <number> words - correct selection - undo that - move up - move down <number> lines
command (**cmd**) + context (**ctx**) + context argument (**ctx-arg**)	- move before <word> - move after <word>
command (**cmd**) + command argument (**cmd-arg**) + context (**ctx**) + context argument (**ctx-arg**)	- insert <word> before <word> - insert <word> after <word> - replace <word> with <word> - change <word> to <word>

Figure 3: Six valid combinations of the four components with example commands for each combination.

4.1 Study Design

Our study comprised two complementary parts. In the *first* observational part, participants used voice commands to correct erroneous sentences while we documented their correction processes, instruction choices, target acquisition strategies, and reactions to the VUI system. We designed four representative text correction tasks based on common error patterns identified in Palin et al.'s [43] mobile typing dataset:

- **T1:** Text correction by *inserting a phrase*
  (e.g., The enforcement has responsibility ... → The ***law*** enforcement has responsibility...)
- **T2:** Text correction by *deleting a phrase*
  (e.g., Was ***is*** it a car wreck? → Was it a car wreck?)
- **T3:** Text correction by *replacing a phrase*
  (e.g., Every employee had an ***internet*** email address. → Every employee had an ***internal*** email address.)
- **T4:** Text correction by *fixing the typo of a phrase*
  (e.g., There were ***freqwuent*** electricity and water shortages. → There were ***frequent*** electricity and water shortages.)

Each participant completed 5 trials per task type, totaling 20 trials presented in randomized order. In the *second* part, we conducted semi-structured interviews to explore participants' thought processes when formulating instructions, their strategies for task completion and error recovery, and their overall experience with the system.

4.2 Study Setup and Apparatus

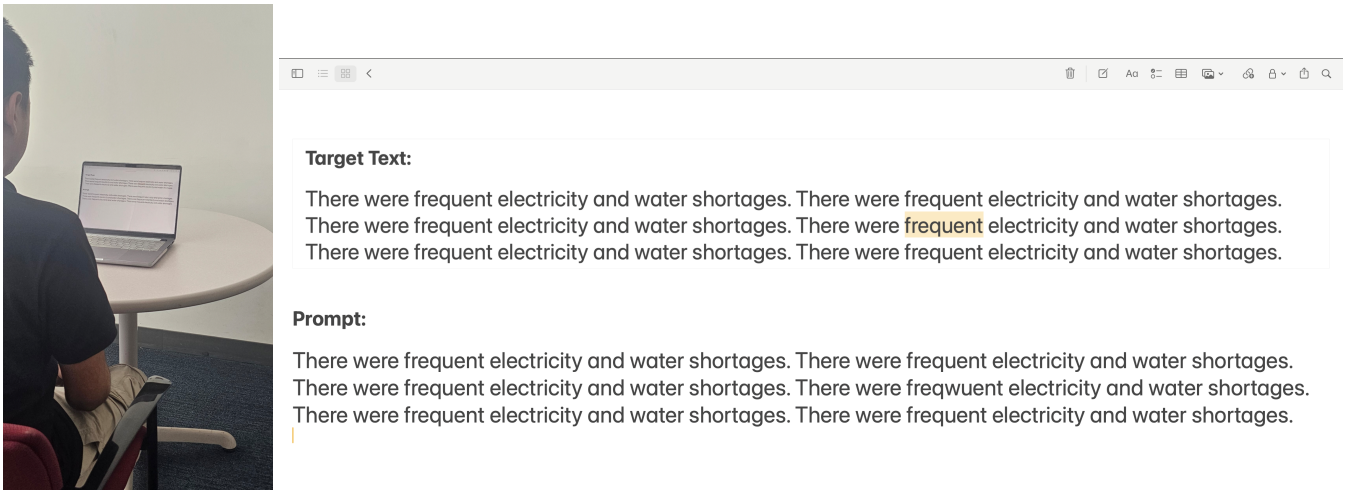
We conducted the study in a quiet room with participants seated before a MacBook Pro running Apple Voice Control (Figure 4a) as the VUI. For presenting correction tasks, we used the Notes application, a popular text editing environment for dictation users. As shown in Figure 4b, each trial displayed both the `Target Text` (correct version with the target word highlighted) and the `Prompt` (erroneous version to be corrected). Each prompt contained exactly one error compared to its target.

To create a realistic correction scenario where errors might appear anywhere within a body of text, we followed Zhao et al.'s [67] approach by embedding each erroneous sentence within a paragraph of 1-7 sentences. These additional sentences—identical copies of the correct version—served as distractors during target acquisition. For example, in Figure 4b, the highlighted word **"frequent"** in the target appears as **"freqwuent"** in the prompt, surrounded by correctly spelled instances of the same sentence.

We deliberately avoided suggesting specific commands for any task, instead observing which instructions participants naturally selected to accomplish each correction.

4.3 Study Procedure

After obtaining informed consent and demographic information, we began with a training session on the VUI system and its supported text correction commands. Participants practiced with these



(a) (b)

Figure 4: Study setup showing (a) the physical arrangement with the participant seated before the laptop running Voice Control and (b) the interface displaying a sample correction task with target text and prompt.

commands until they reported feeling comfortable with the system. We then provided sample trials for each task type to ensure familiarity with the correction process.

For the main study, we presented 20 trials in randomized order to each participant. The randomization was performed at two levels: (i) task assignment was randomized such that participants received correction tasks (insertion, deletion, replacement, and typo correction) in varying sequences rather than grouped by type, and (ii) within each task type, the specific trial instances were randomized. This ensured that no two participants experienced the same task sequence and that learning effects were not confounded with specific task types.

In each trial, participants first reviewed the `Target Text` and `Prompt` to understand what and where to correct. Once they acknowledged understanding the required correction, we switched to another Notes tab containing only the `Prompt` text. Participants could revisit the target if needed, and could use as many commands as necessary until the text matched the target. We advanced to the next trial only when participants confirmed completion of the current one, with rest periods available between trials. Following the correction tasks, we conducted a semi-structured interview to explore participants' specific strategies and experiences.

4.4 Data Analysis

We transcribed and analyzed the data through iterative coding [5], with all authors participating in weekly research meetings to refine the codebook, identify key concepts, organize categories, and resolve disagreements. Our initial coding cycle identified low-level descriptive codes representing specific command and task strategies. In subsequent cycles, we merged conceptually similar codes into broader categories that revealed primary themes. This refinement process ultimately yielded four key themes that we describe next.

4.5 Study Findings

Our formative study revealed critical usability challenges with fixed-format voice commands that inform the design of VoiceAlign. We organize these findings around four themes: target acquisition, instruction planning and execution, task completion strategies, and overall system usability.

4.5.1 Target Acquisition Challenges. We found that target acquisition with voice commands was cumbersome and inconvenient in two cases: when the target had multiple occurrences or contained a typo/out-of-vocabulary words. If participants used a `select <phrase>` command where the `<phrase>` occurred multiple times, the system considered all occurrences as the potential target and tagged each of them with a numeric index starting from 1. Then participants issued additional `choose <number>` or just the `<number>` command to disambiguate the particular `<phrase>` they wanted to select. All participants reported this process to be time-consuming. P15 thought it was extra decision-making for him as he had to look for the exact instance out of the duplicates he wanted to select. P11 liked numbers for disambiguation but also wanted to have the numbers during selection, not only during disambiguation.

When the target was a typo or an out-of-vocabulary word, participants had difficulty acquiring them as the ASR did not recognize them. The first strategy that all participants tried was to utter the typo as a word, if possible, by dividing it into syllables. For example, they uttered the typo 'cintrol' as a two-syllable word 'cin'–'trol' and used that as the `cmd-arg`, but the ASR could not recognize it. The most common and successful strategy participants employed to select a typo required three commands in most cases: (i) use a `select` command to select a word adjacent to the typo; (ii) use a `choose` command to disambiguate (if duplicate); and (iii) another `select` command with `previous/next word` as `ctx` to select the typo. Another option was to spell out the typo using the `spelling mode`. However, spelling longer words containing typos (e.g., 'freqwuent' containing 9 characters) was prone to misrecognition because if the

system could not recognize a single character, the selection failed. Therefore, participants needed multiple attempts with longer typos.

For target acquisition, participants preferred using the `select` command predominantly over the `move` command. In addition, they also preferred easier `ctx` over larger ones, such as `select previous word` over `select previous <number> words`. We never encountered commands such as `move left <number> words`, which required a distance calculation. The only `move` commands participants used were `move before/after <phrase>`. In other words, selection-based target acquisition was more convenient than navigation-based acquisition, which is also corroborated in prior work [48].

4.5.2 Command Planning and Execution Barriers. Successfully executing voice commands required participants to select a valid combination of command components, retrieve each component's value from short-term memory, and complete the utterance before timeout.

Initial attempts frequently failed because participants did not plan their utterances to match the system's expectations. With four possible components creating 16 potential combinations — only 6 of which are valid (Figure 3) — participants faced a 40% chance of selecting a valid combination. This probability decreased further as different commands expected specific component combinations.

Participants described commands as "restrictive," "programming-like," and "unnatural." They struggled with several structural issues, including component quantity mismatches, synonymous commands, context word interchanging, and natural speech patterns. As P3 observed: *"It is not natural; it is not the way we speak. It is like we need to change ourselves. It is like you have to follow very specific instructions."*

Even when participants correctly structured their commands, timeouts frequently disrupted execution. The process of retrieving component values from different sources while maintaining the precise utterance pace proved challenging. This difficulty intensified for complex commands requiring context words that established relationships between arguments.

P2 articulated this challenge: *"I have to say the exact command... At the same time, I have to also say it fast... I have to process my sentence to make up the command, and that too in a precise time. That's a bit tricky!"*

First attempts typically failed due to attention division between component retrieval and coherent utterance. Most participants initially required three attempts per command (spiral depth of three), eventually improving to two attempts with practice. Unsurprisingly, timeout frequency increased with command complexity, leading most participants to prefer simpler two-component commands.

We analyzed the uttered commands from our study that did not work and categorized them into 8 categories. Table 2 shows these categories, the incorrect commands, and their correct versions. Notice that the incorrect commands are minor variations of the correct ones that participants thought made sense and would give flexibility and ease-of-use if supported by the system.

4.5.3 Task Completion Strategies. The open-ended nature of voice commands required participants to strategize their approach before execution. We observed several consistent patterns:

Participants instinctively began with target acquisition across all tasks, mirroring desktop interaction patterns where selection precedes action. However, selected targets did not persist between commands, creating redundant work. For example, when inserting text near a selected and disambiguated location, the system would require re-disambiguation after the insert command, rendering the initial selection meaningless.

Over time, participants developed a consistent two-step strategy of acquisition followed by correction, both using two-component instructions. For example, they would select a word and then use a deictic reference like `delete that` to modify it. When commands didn't support such references (as with `insert` and `replace`), participants resorted to workarounds like deleting targets and re-dictating replacements.

When facing errors, participants employed various recovery strategies: using `undo that`, switching to `command mode`, or using `spelling mode` for unrecognized words. Most notably, once participants found a successful approach — even if suboptimal — they adhered to it, following an "if it works, don't break it" strategy to minimize errors.

We analyzed how participants completed the four correction tasks in-depth and identified how they performed the same correction tasks in different ways. We outlined the seven different ways participants completed the insertion task in Figure 5 along with a visual representation of how each command modifies the text through Voice Control. Notice the persistence of selection at the beginning that does not benefit due to additional disambiguation. Additionally, the insertion was often done through other commands that participants kept using if successful.

4.5.4 System Usability Issues. Beyond specific command challenges, participants identified several broader usability concerns. They reported having to proactively control the microphone to prevent unintended dictation, particularly when thinking aloud or reading text to themselves. This often led to cascading errors when expressions of frustration themselves became dictated text.

Participants expressed mixed preferences regarding correction modes, with most seeking to minimize mode switching. While half the participants valued both dictation and command modes equally, others preferred one over the other based on flexibility and error prevention needs. The `spelling mode` was generally considered a last resort despite its utility for specific tasks.

All participants highlighted insufficient system feedback as a major issue. They often couldn't determine whether an uttered command had been recognized, whether a timeout had occurred, or which mode was currently active. As P7 suggested, the system should "clearly distinguish between just listening and listening to a command" through visual cues like live transcription of recognized command components.

4.6 Discussion: Designing Better Voice-Based Text Correction

Our findings reveal a fundamental mismatch between legacy VUI command structures and users' cognitive models of speech interaction. Unlike programming, which is primarily text-based, speech is naturally more verbose, erroneous, and disorganized. The rigid instruction formats fail to accommodate these characteristics, creating a substantial cognitive burden.

Category	Description	Incorrect Command	Correct Command
Swap `cmd`	Swapping one valid `cmd` for another valid one, particularly common for select and choose.	`select <number>`	`choose <number>`
		`choose <phrase>`	`select <phrase>`
Substitute `cmd`	Substituting one valid `cmd` with another invalid synonym.	`add <phrase> before <phrase>`	`insert <phrase> before <phrase>`
		`fix <phrase>`	`correct <phrase>`
		`remove <phrase>`	`delete <phrase>`
Substitute `ctx`	Substituting one valid `ctx` with another invalid synonym.	`select left word/select word before`	`select previous word`
		`select right word/select word after`	`select next word`
		`replace <phrase> to/using <phrase>`	`replace <phrase> with <phrase>`
Substitute template	Substituting one valid template with another template with more or fewer components.	`insert <phrase>`	`insert <phrase> before <phrase>`
		`delete <phrase> before <phrase>`	`delete <phrase>`
Ignore deictic args	Ignoring deictic arguments such as this/that after selection or for commands that do not have any particular reference.	`delete`	`delete that`
		`undo/redo`	`undo that/redo that`
Add deictic args	Adding deictic arguments such as this/that after selection in four-component commands.	`insert <phrase> before that`	`insert <phrase> before <phrase>`
		`replace that with <phrase>`	`replace <phrase> with <phrase>`
Missing args	Missing arguments in four-component commands after selection to reduce command complexity.	`insert <phrase> before`	`insert <phrase> before <phrase>`
		`replace with <phrase>`	`replace <phrase> with <phrase>`
Natural utterance	Uttering fixed-format commands naturally, creating minor variations.	`choose number <number>`	`choose <number>`
		`correct the selected word`	`correct that`

Table 2: Categories of incorrect commands identified during the study and their correct versions. Note that all uttered commands are minor variations of the fixed-format command templates, which are not supported by VUIs.

Based on our study, we propose six design guidelines for improving speech-based text correction:

(1) **Simplify target acquisition** by extending numeric indexing to all words in text, allowing direct selection through `select <number>` commands.
(2) **Prioritize two-component instructions** to reduce cognitive load, with systems inferring contextual relationships (like insertion position) based on text analysis or user preferences.
(3) **Maintain selection context** across commands, preserving target information from previous operations to eliminate redundant selections.
(4) **Minimize timeouts** by extending thresholds and implementing user-defined completion phrases like "over" or "end" to signal command completion.

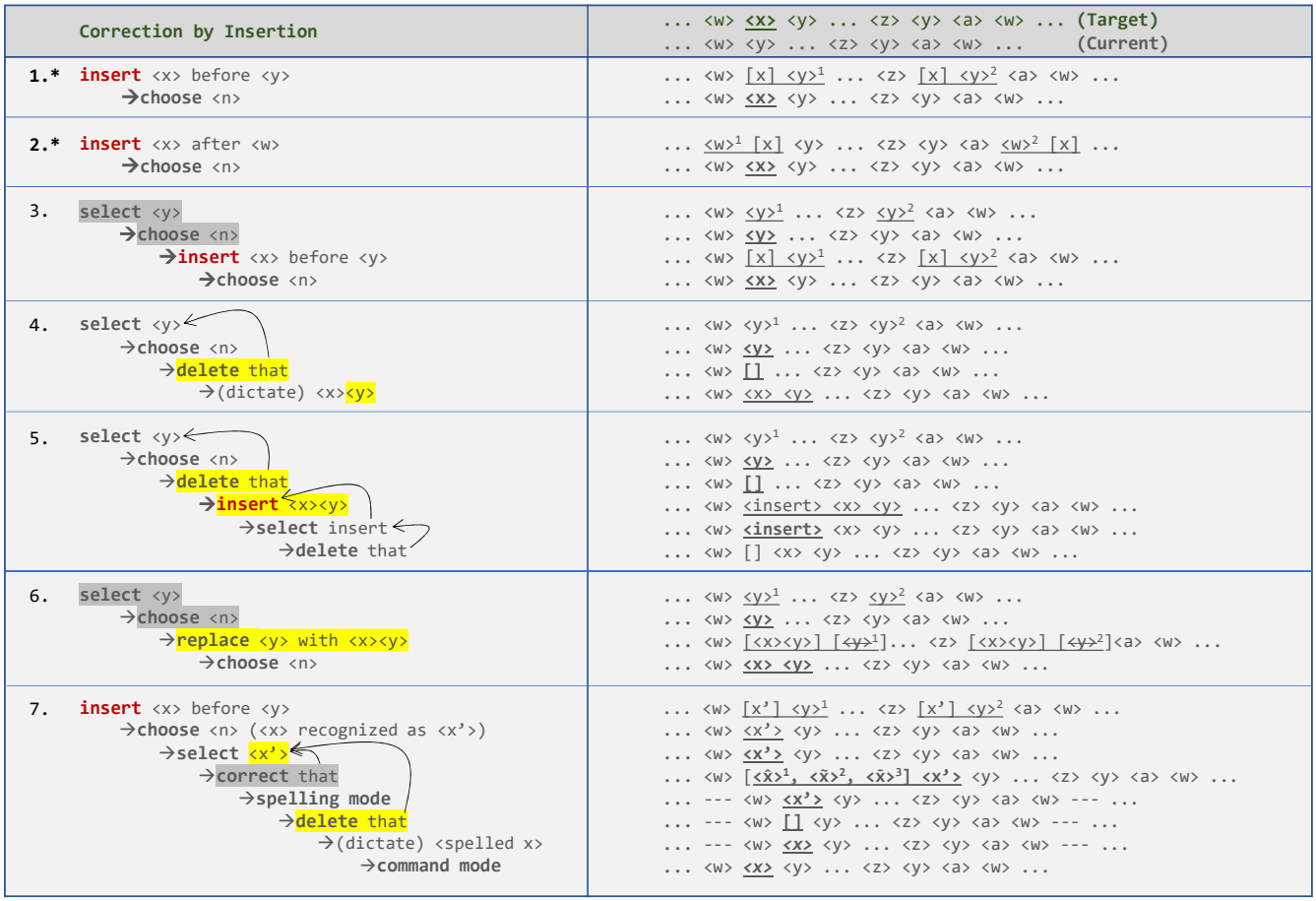


Figure 5: Common instances of command sequences used by participants to accomplish corrections by insertions on the left and the workflow of the system on the right. An asterisk (*) indicates the optimal command sequences. The target text and the current state are shown at the top right. An underline is used to indicate where editing occurred. A square brace ([]) indicates a temporary buffer the system uses internally (e.g., all potential insertions until disambiguated).

(5) **Support minor variations** in commands through more flexible interpretation, potentially leveraging large language models (LLMs) to understand intent despite syntactic variations.
(6) **Enhance system feedback** through clear visual and auditory cues indicating recognition status, active mode, and detected command components.

These guidelines inform our development of VoiceAlign, a shimming layer that addresses these limitations while preserving compatibility with existing legacy VUI systems.

5 Designing VoiceAlign: An Adaptive Shimming Layer

Our analysis of voice commands indicates that legacy VUIs provide a comprehensive set of commands for interaction and text correction. However, our formative study revealed that users face difficulty using these comprehensive command sets due to fixed formats, timeouts, and insufficient feedback. These systems are black-box and difficult to modify, while creating a completely new VUI with all the functionality that existing VUIs support is costly and cumbersome. Therefore, in this work, we aimed to leverage the capabilities of existing VUIs while also allowing users to speak command variations we identified during our formative study without modifying them.

This led us to design VoiceAlign, an adaptive shimming layer that mediates between users and legacy VUI systems. The concept of shimming layers originated in software engineering as the Adapter pattern [13]. VoiceAlign has the following design goals:

- **Interfacing with Existing Black-box VUIs**: Existing VUIs, such as Voice Control, are widely available and offer comprehensive features. We aim to interface with these systems, treating them as black boxes, allowing us to leverage their features.
- **Supporting Command Variations**: We aim to support command variations, allowing users to speak commands naturally and align them with fixed-format commands using large language models.

- **Enhancing User Experience**: Our goal is to enhance user experience by providing additional feedback to keep users informed while using voice commands and giving them sufficient time for planning and uttering, as we identified during our formative study.

5.1 System Architecture and Workflow

VoiceAlign has two primary components: a command adapter and a command whisperer (see Figure 1). Currently, it runs as a web application in a browser.

5.1.1 Command Adapter. The command adapter intercepts voice input before it reaches the legacy VUI system. This component serves three functions:

- **Command recognition**: It captures and transcribes user utterances in real-time using Web Speech API[1] and provides immediate visual feedback on detected speech.
- **Timeout management**: It implements a customizable inter-/intra-component timer (initially set to 3 seconds), giving users sufficient time to plan and articulate commands without pressure.
- **Command transformation**: Using an LLM (Claude3.5-Sonnet API, with temperature=0), it maps users' natural language commands to syntactically valid formats required by the legacy system. For each command, the LLM receives: the user's transcribed utterance; the currently selected text (if available); and a history of the five most recently executed commands. It then extracts necessary components (`cmd`, `cmd-arg`, `ctx`, and `ctx-arg`) and reformulates them into VUI-specific syntax.

The system maintains contextual awareness by tracking selection state through command history. When users issue a `select <phrase>` command, VoiceAlign stores it in its internal cache. For commands like `choose <n>`, it references command history to determine whether to preserve the current selection, mimicking Voice Control's default behavior.

When a user speaks, the adapter displays an active microphone icon indicating active listening, shows real-time transcription, and processes the input once the timer expires (see Figure 6).

5.1.2 Command Whisperer. The command whisperer creates a virtual audio channel connecting the adapter to the legacy VUI through two components:

- **Virtual channel creation**: It establishes a loopback audio pathway using BlackHole[2] that redirects system audio output to serve as input for the legacy VUI.
- **Text-to-speech relay**: Once the adapter produces a corrected command, the whisperer converts it to speech using the system TTS engine and plays it through the virtual channel to the legacy VUI.

This approach allows VoiceAlign to function as a transparent layer, with the legacy VUI unaware that commands are coming from an intermediary rather than directly from the user.

[1]https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API/Using_the_Web_Speech_API

[2]https://github.com/ExistentialAudio/BlackHole

5.2 LLM Prompting Strategy

Our implementation uses a carefully crafted prompting strategy to ensure reliable command transformation while avoiding hallucinations or fabricated commands that could disrupt the user experience. We developed the prompt iteratively based on users' utterances and logs from the formative study, particularly those that resulted in errors (§4), as well as our generalized command template developed during the analysis of legacy VUIs (§3).

5.2.1 Role Definition and Command Analysis. We designed a role-based prompt where the LLM functions as an "advanced command corrector" specialized in text editing operations. The prompt instructs the model to:

(1) Analyze the command to identify its intended operation type.
(2) Determine the required components for that operation.
(3) Extract arguments directly from the user's utterance without inference.
(4) Reconstruct a syntactically valid command only when all required components can be reliably extracted.

To prevent command fabrication, we explicitly constrain the model to use only consecutive words directly from the user's utterance for arguments, prohibiting the generation of new content except in specific, well-defined cases.

5.2.2 Command Completion and Error Handling. For commands that reference previous selections (e.g., `insert <phrase> before that`), we provided explicit guidelines for resolving deictic references based on selection history. This enables VoiceAlign to support natural command sequences while maintaining syntactic precision.

When a command lacks sufficient information for reliable transformation, the system shifts to suggestion mode rather than attempting correction. As shown in Figure 6 (right), it provides structured guidance on how to complete the command, maintaining user trust by avoiding potentially incorrect transformations.

5.2.3 Confidence Assessment. To further enhance reliability, our prompt requires the model to:

- Show explicit reasoning for each transformation step.
- Assign a confidence score (0-100) to each correction attempt.
- Only apply corrections with high confidence scores.

This approach, inspired by Li et al.'s work on improving LLM precision [27], ensures that VoiceAlign prioritizes accuracy over coverage, preserving user trust in the system's transformations.

6 Evaluation of VoiceAlign

We conducted a summative study to evaluate the effectiveness and user experience of VoiceAlign when working as a shim layer between users and VUIs. This section describes the study setup, findings, and comparisons with the earlier study.

6.1 Participants, Study Design, and Setup

We recruited 12 participants (10 males, 2 females, ages 26-32) using the same recruitment criteria as our formative study. Participants, anonymized as P1 through P12, performed identical text correction tasks using VoiceAlign: 20 erroneous sentences across four task types. The study setup mirrored our formative evaluation with one

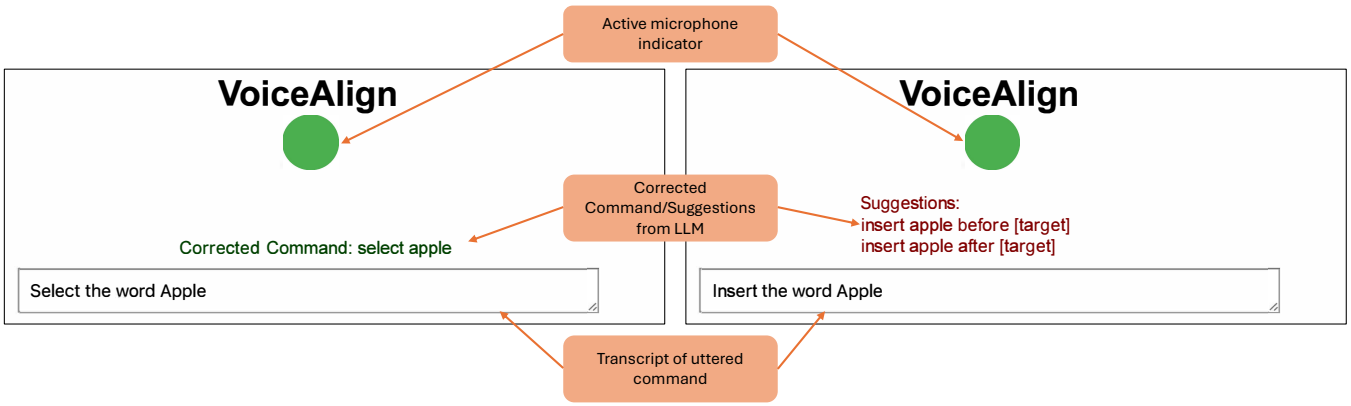


Figure 6: VoiceAlign interface containing an indicator when the microphone is active, an input box to display the transcript of users' uttered commands in real-time, and the output from the LLM providing the correct command or a list of suggestions. (Left) An example of a command correction, where `'Select the word Apple'` is transformed to the syntactically valid `'select apple'` by removing extraneous words while preserving core components. (Right) An example of a command suggestion, where `'Insert the word Apple'` lacks required context parameters, prompting the system to offer structured guidance on how to complete the command.

key difference: participants could now see the VoiceAlign interface displayed above the Notes application, providing supplementary feedback alongside Voice Control's native responses.

6.2 Study Procedure and Data Analysis

After collecting consent and demographic information, we introduced participants to the VoiceAlign interface and provided them with Voice Control commands.

Participants completed 20 trials of the four correction tasks in random order. Again, the randomization was performed at both the task level (trials for the tasks were randomized) and within each task type (the specific trials were randomized) to mitigate the learning effect.

Rather than instructing participants to deliberately vary their command structures, we observed their natural interactions with Voice Control, noting incorrect commands, VoiceAlign's responses, and participants' subsequent behaviors.

Following the correction tasks, we conducted semi-structured interviews and administered the NASA-TLX questionnaire to measure task load. Each session lasted 60-75 minutes, was video-recorded for analysis, and concluded with participants receiving a $20 Amazon gift card as compensation. Our data analysis followed the same approach used in the formative study.

6.3 Quantitative Results

6.3.1 Number of Commands per Task. As shown in Figure 7a, VoiceAlign reduced the average number of commands needed per correction task by 25% — from 4.92 (SD: 0.51) with Voice Control to 3.67 (SD: 0.48) with VoiceAlign + Voice Control. An independent samples t-test confirmed this difference was statistically significant ($t(26) = 6.56, p < .001$). By preventing command failures, VoiceAlign enabled participants to complete tasks with ease.

6.3.2 Command Failures. Figure 7b illustrates the percentage of failed commands across systems. With Voice Control alone, 26.90%

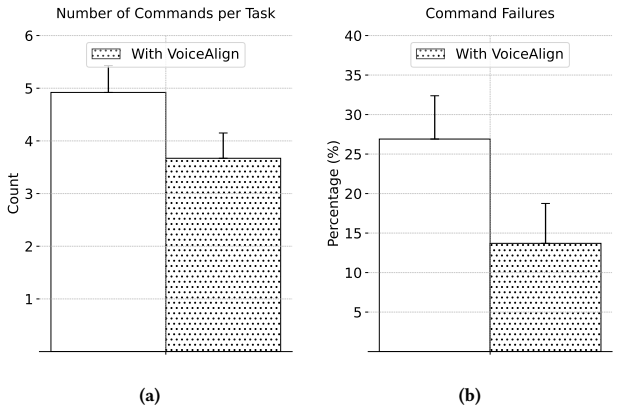


Figure 7: (a) Average number of commands participants used to perform each text correction task. (b) Percentage of commands that failed out of all commands by participants.

(SD: 5.48%) of commands failed, compared to 13.70% (SD: 5.05%) when VoiceAlign adapted commands before passing them on to Voice Control. An independent samples t-test confirmed this difference was statistically significant ($t(26) = 6.526, p < .001$).

We further analyzed the command failures for Voice Control with and without VoiceAlign. We categorized the failures from Voice Control without VoiceAlign into three categories: timeouts, incorrect command syntax, and recognition errors. Notably, timeouts caused 33% of command failures in Voice Control. 35.8% of the failures were due to incorrect command syntax, shown in Table 2, and the rest of the commands failed due to misrecognition.

For VoiceAlign + Voice Control, the majority of the command failures (76.5%) were due to incorrect recognition. Approximately 20% of the failures occurred due to incorrect command syntax, which VoiceAlign could not correct due to missing arguments. However, in

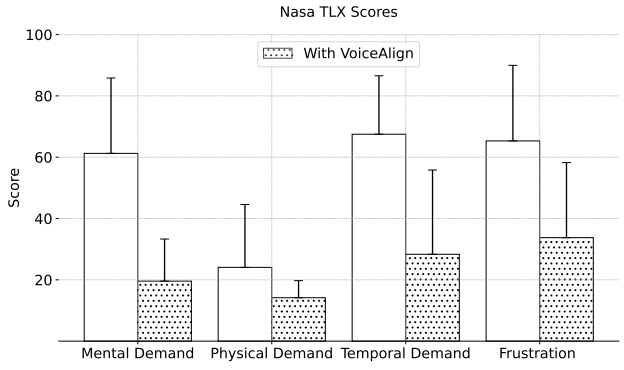


Figure 8: Comparison of NASA-TLX Scores with and without VoiceAlign.

such cases, users were informed as VoiceAlign provided suggestions to the users, which they could use in the next attempt to correct their command. No timeouts occurred with VoiceAlign due to our adjusted timeout mechanism in the shimming layer. VoiceAlign could commonly correct commands with swapped or substituted `cmd` or `ctx`, natural utterances, and ignored/added deictic args (Table 2). However, we noticed a few occurrences where VoiceAlign provided a suggestion for correctly uttered commands containing 'that' as an argument (e.g., `correct that`) without prior selection, where the model suggested selecting first.

6.4 Subjective Findings

6.4.1 NASA TLX Score. The NASA-TLX scores (Figure 8) revealed that VoiceAlign reduced mental demand, physical demand, temporal demand, and frustration when paired with Voice Control. A Mann-Whitney U test confirmed significant reductions in mental demand ($Z = -3.692, p < .001$), temporal demand ($Z = -3.180, p = .001$), and frustration ($Z = -2.687, p = .007$), though the reduction in physical demand was not statistically significant ($Z = -1.448, p = .148$).

6.4.2 VoiceAlign Adapted Voice Commands Successfully. All participants responded positively to VoiceAlign's automatic command correction. When P1 accidentally issued `select <n>` instead of `choose <n>` and saw the command automatically corrected, he exclaimed: *"Awesome! That's how it should be!"*. We observed similar enthusiasm across participants when they realized they could deviate from exact command formats while still achieving their goals.

Some participants adapted their command strategies based on VoiceAlign's capabilities. For example, P5 discovered that after selecting a word, she could refer to it in subsequent commands using deictic references (e.g., `replace that with <phrase>`). She commented:

> *"This makes more sense to me. I can use the same strategy for other commands. I can select a word and then say delete that. Then why can't I say 'replace that with [<phrase>]'? It helps me because I do not have to think about two words for one command. I would rather replace that than say two words."*

This finding aligned with our formative study, where participants reported that combining multiple arguments in a single command increased mental load. VoiceAlign also successfully handled synonymous commands (e.g., `fix that` for `correct that`) and removed verbal fillers and false starts (e.g., "Okay...`select <phrase>`").

We observed a few edge cases where VoiceAlign incorrectly discarded valid commands because the LLM lacked sufficient context, particularly with deictic references following selection commands that modified context in ways the LLM couldn't track.

6.4.3 VoiceAlign Mitigated Timeout Errors. All participants successfully issued commands within our extended 3-second timer, eliminating the timeout errors. None reported difficulties planning or articulating command components, which corresponded with significantly decreased mental and temporal demand scores.

Two participants (P4 and P12) suggested that the timer could adaptively adjust based on user expertise, allowing longer intervals for novices and shorter ones for experienced users. P4 noted that for multi-component commands, the system could potentially use syntactic cues to recognize completion: *"If I already said more than four words, chances are I am done with my command, so the system does not need to wait as much."* Both appreciated that the timer was customizable.

6.4.4 VoiceAlign Kept Users Informed through Detailed Feedback. All participants valued the real-time transcription and feedback provided by VoiceAlign. P2 appreciated seeing how the system recognized his commands, noting that transcription errors helped him prepare alternative approaches. P12 found the microphone indicator particularly helpful for understanding when he could and couldn't issue commands.

While the suggestions provided when commands couldn't be corrected proved useful, we noted that the LLM sometimes suggested commands based on syntactic viability rather than perceived user intent. For example, it might suggest a delete command when the user was attempting to insert text. Participants occasionally found these misaligned suggestions distracting.

6.4.5 Perception of Time Delay. Half of the participants noted the latency introduced by LLM processing and text-to-speech conversion. Particularly, the API response time depends on the internet connectivity and can take up to 2-3 seconds. P7 commented that while the delay was acceptable for occasional edits, it could become problematic during extended editing sessions requiring many consecutive commands.

7 Reducing Response Latency of VoiceAlign

While VoiceAlign improved user experience by allowing users to speak commands naturally and correcting the commands to the fixed-format, the time required for LLM API calls also introduced latency. In addition, depending on external APIs has cost implications and requires stable internet connectivity.

To further improve the system, we aimed to reduce response latency and run the model locally to eliminate costs and connectivity issues. Therefore, we fine-tuned a small language model, Gemma 3, with 270 million parameters, to convert users' commands to VUI

format and served the fine-tuned model locally through Ollama[3]. The fine-tuning process is informed by our formative study. We describe the process and the results in this section.

Dataset Generation for Fine-Tuning. We generated synthetic fine-tuning data using a large language model, creating separate datasets for training, validation, and testing. We started with the commands collected during our prior studies and created incorrect and correct command pairs. We then provided these pairs to Claude Sonnet 4.5 along with the command templates (Figure 2), example commands (Figure 3), and the incorrect command categories (Table 2), prompting it to generate samples for each command pair, both correct-to-correct and incorrect-to-correct mappings. We also generated samples where incorrect commands have missing information, requiring further clarification from users.

We reviewed the generated samples, corrected any discrepancies, and provided the corrected examples back to Claude to refine the command correction process. We iterated this review-and-correction cycle until the generated samples consistently matched our quality standards. We then prompted the model to generate three synthetic datasets by varying parameters, substituting synonymous commands, and adding natural phrases (e.g., "can you please...", "I want to...") observed during our prior studies. Our final dataset consisted of 1,000 training samples, 400 validation samples, and 150 test samples.

Each sample contains a fixed task (passed as the system prompt: *Convert the following natural language command to the correct voice control command format.*), an input, and an expected output. For commands that are correct or contain all required information, the input includes the utterance and the current selection (which we maintain and update following each select command), and the expected output is the correct command expected by Voice Control. For commands with missing required information, the expected output is a follow-up question, which is concatenated to the input when the user responds. Example inputs and expected outputs are outlined in Table 3.

Our synthetic dataset covers all six valid command structure combinations from Figure 3, with balanced representation across 8 command operations from Table 1, which is shown in Figure 9a (`correct`: 18.5%, `select`: 17.6%, `replace`: 16.9%, `insert`: 13.7%, `delete`: 11.6%, `choose`: 9.8%, `undo`: 6.2%, `redo`: 5.8%). The dataset captures all error categories from Table 2, which is shown in Fig. 9b: substitute template (38.5%), natural utterances (21.3%), substitute cmd (17.7%), missing args/added deictic/ignored deictic args (15.4%), and substitute ctx (7.1%). We incorporated 50+ synonym variations informed by our formative study. The training samples represent 195 distinct correct commands, with multiple natural language variations for each (5.1 variations per command on average). This high variation density ensures the model learns to adapt diverse user inputs to correct command syntax rather than memorizing fixed patterns. The dataset balances commands with (29.2%) and without (70.8%) selection context, reflecting the full range of complexity observed in our formative study.

Fine-tuning Process. We fine-tuned the Gemma 3 270M instruction-tuned model using parameter-efficient fine-tuning with LoRA [21] (r = 128). Training was performed on an NVIDIA RTX A6000 GPU using the Unsloth framework [10] for efficient fine-tuning. We used the SFTTrainer with AdamW optimizer, with approximately 10% of the model's parameters being trainable. Validation loss was calculated after each epoch, and the model was saved accordingly.

Evaluation. We evaluated the model before and after fine-tuning on the test set using exact match accuracy and ROUGE-L score [28] for semantic similarity. Before fine-tuning, the model achieved an exact match accuracy of 8.05% and a ROUGE-L score of 0.2791 with zero-shot prompting. With 5-shot prompting, the accuracy and the ROUGE-L score slightly increased to 11.82% and 0.3145, respectively.

After fine-tuning, the model achieved 90.6% exact match accuracy and a ROUGE-L score of 0.9587. Upon closer examination of mismatches, we observed that the model occasionally asked clarifying questions in rare cases where commands were ambiguous. One such example is "transform work" where "transform" is used as a synonym of "replace" and the expected output asks "what should I replace work with?" The model instead asked "what should I correct?" However, this example represents an edge case with an uncommon synonym that never appeared in our user studies, suggesting the model performs well on realistic user inputs.

We converted the fine-tuned model into GGUF format and served it using Ollama to measure inference performance on a MacBook Pro. The model responded within 200 ms, with 95% of responses completing in 130-150 ms. This demonstrates that the fine-tuned model enables real-time interaction on local devices while eliminating API latency and costs.

Overall, we found that smaller models fine-tuned for specific tasks informed by user studies can be accurate, fast, and effective for local on-device inference, providing greater flexibility for voice command systems. We plan to release the data and code in the future.

8 Discussion

Our studies revealed consistent patterns in user behavior. Participants instinctively selected targets before performing editing operations, even though this approach required multiple disambiguations. When legacy systems discard the context of earlier selections, they miss an opportunity to align with users' mental models and create frustration. While we partially mitigated this issue by extending select and replace commands, the black-box nature of these systems prevented a complete solution.

Our participants struggled with four-component commands, which demanded significant attention. Providing sufficient time and reducing commands to two components substantially improved their experience. We also observed that participants unconsciously reverted to synonyms even after mastering the correct commands. This finding emphasizes the importance of supporting minor variations in command phrasing while preserving semantic intent.

8.1 Supporting Human-AI Interaction in VUIs

Our work with VoiceAlign yields insights that extend beyond adapting legacy VUI systems to inform broader questions in human-AI (HAI) interaction, particularly regarding how AI can mediate between natural human communication and rigid system constraints.

[3]https://ollama.com/

Sample Type	Input	Expected Output
Exact command	select previous word | selection: apple	SELECT PREVIOUS WORD
Natural utterance	can you please select the next word | selection: apple	SELECT NEXT WORD
Swap `cmd`	choose the word meeting | selection:	SELECT meeting
Substitute `cmd`	fix meeting | selection:	CORRECT meeting
Substitute `cmd`, Natural utterance	please add at home before that | selection: tonight	INSERT at home BEFORE tonight
Missing arg	insert before apple pie | selection:	ASK: What should I insert before apple pie?
Missing arg (follow-up clarification)	insert before apple pie | selection: | CLARIFICATION QUESTION: What should I insert before apple pie? | CLARIFICATION: in the morning	INSERT in the morning BEFORE apple pie

Table 3: Sample input and expected output from the synthetic dataset.

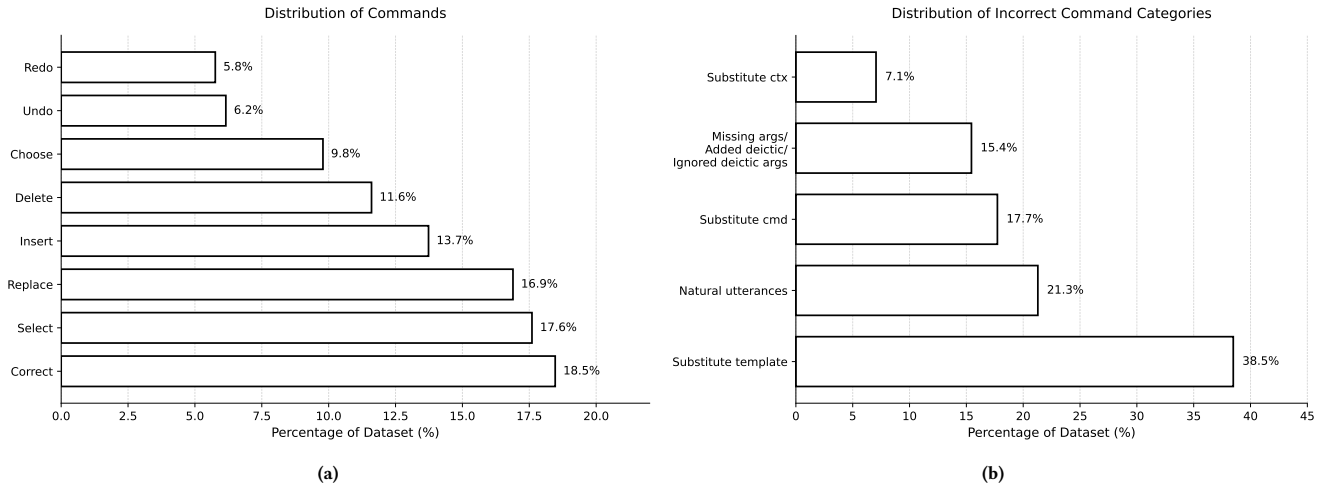


Figure 9: Distribution of (a) text correction commands and (b) incorrect command categories in the synthetic dataset.

8.1.1 Balancing Human Planning Time with AI Response Speed. A fundamental issue in voice-based human-AI interaction is providing users sufficient time for cognitive processing while maintaining real-time responsiveness. Users need time to plan multi-component utterances, retrieve arguments from memory, or visually locate information, yet they expect immediate system feedback characteristic of natural conversation. AI-mediated voice systems must balance these opposing needs while providing distinguishable signals: indicators showing when the system is listening versus processing, real-time transcription maintaining users in the interaction loop, and minimized AI latency that preserves the conversational flow. This temporal design challenge is particularly acute for shimming layers augmenting legacy systems where users have internalized expectations of fast response times. Therefore, any AI mediation must feel seamless rather than introducing perceptible lag that disrupts the natural dialogue experience.

8.1.2 Context Retention and Progressive Collaboration to Minimize Repetition. Speech is inherently conversational and context-dependent, yet rigid command structures that require repetition and discard context undermine voice as a natural interaction modality. AI systems should function as conversational partners that maintain context across turns and scaffold incomplete input through targeted clarification rather than forcing users to restart. When users provide partial commands, AI should retain available information and ask minimal clarifying questions that allow users to continue their train of thought rather than reconstruct entire commands. This context awareness extends to leveraging information from immediate prior turns — understanding deictic references, implicit arguments, and incomplete commands that characterize natural speech. Overall, AI must adapt to human conversational norms rather than requiring humans to adapt to system constraints, which can lead to a fluent HAI interaction experience.

8.1.3 Small Specialized Models for Real-Time Structured Tasks. Our findings indicate that fixed-format voice commands create conflicting requirements that small, specialized AI models can uniquely resolve. Users cannot reliably produce syntactically perfect commands spontaneously — natural speech contains variations, false starts, and fillers — yet the real-time responsiveness that makes voice feel natural precludes reliance on large general-purpose models that introduce latencies and external dependencies. For systems with predictable command structures and bounded vocabularies,

local fine-tuned models can handle natural variation (synonyms, reordering, disfluencies) while maintaining faster responses. Therefore, when interactions involve mapping natural variation to known formal structures within bounded domains, small specialized models can provide reasonable accuracy, real-time performance, privacy through edge deployment, cost efficiency, and personalization opportunities compared to large general models, enabling voice interfaces that feel both natural and responsive.

8.2 Extending VUI Systems through VoiceAlign

Emerging agentic interaction frameworks like the Model Context Protocol (MCP) [46] represent promising directions for AI-system integration. However, they face fundamental limitations in legacy computer control scenarios. MCP-based approaches require direct API access to application DOM structures and system internals — capabilities that are increasingly restricted in modern operating systems like macOS for security reasons. Legacy VUI systems like Voice Control bypass these restrictions through deep OS-level integration with accessibility APIs, providing comprehensive system control that external frameworks cannot replicate easily. Our shimming layer preserves this valuable low-level access of VUIs while adding an adaptability layer on top, offering a practical middle ground between complete system replacement and acceptance of rigid command structures.

While we focused on text correction commands, our approach of decomposing voice commands into discrete components and analyzing how users combine these components to match command templates applies broadly to other commands supported by fixed-format VUIs, such as computer control commands (`click <icon>`, `open <application>`). Our methodology of breaking down seemingly simple commands into granular components and determining their interrelationships and usability can extend to any VUI command category. Commands with certain characteristics are particularly prone to failure: those requiring many components uttered sequentially, those requiring users to search for or identify information mid-speech, and those referencing contextual information increase cognitive load and failure rates. In such cases, breaking commands into smaller steps, minimizing required components, reducing component variations (e.g., variable component ordering or differing template lengths), and standardizing valid command structures can improve predictability. Therefore, for any VUI system, identifying a comprehensive command template, enumerating valid component combinations, analyzing component interconnectivity, and determining how each component is acquired during utterance can reveal complexity patterns and guide better voice command design.

Our fine-tuning of a local LLM keeps system commands intact while supporting minor command variations to reduce cognitive demand and command retries. This approach generalizes to other VUI systems by generating synthetic datasets with command variations and fine-tuning local models for those systems. The fine-tuning methodology also enables personalization, as not all commands need adaptation. Users could select their most frequently used commands and provide example variations, allowing an LLM to generate a personalized synthetic dataset that enables our shim layer to adapt variations most relevant to individual workflows, leading to better usability.

Beyond command adaptation, our VoiceAlign architecture can address other VUI limitations. For multilingual support, our approach could help users interact with English-only VUIs using their preferred languages. Recent LLMs demonstrate strong translation capabilities [47, 65], potentially allowing users to speak commands in their native language before VoiceAlign translates them into English and relays them to the legacy system. Similarly, this approach could improve VUI performance in noisy environments by preprocessing commands to separate the signal from noise [61, 63]. The command adapter could filter background interference and extract critical command components before sending a clean, structured command to the underlying system.

8.3 Privacy, Trust, and Transparency in AI-Mediated VUI Interaction

Privacy, trust, and transparency have been longstanding concerns for VUIs, particularly with the integration of AI [1, 26]. When voice commands are processed through cloud-based LLM APIs, sensitive user data, including document content, personal communications, and contextual information, is transmitted to external servers where it may be retained, aggregated, or used for model training without explicit user consent [32]. This practice constitutes a potential breach of user privacy, and the opacity of data handling practices by API providers undermines transparency [4]. Prior research has shown that VUI users are often unaware of how their data is collected and used, placing trust in systems to protect their privacy. However, users with heightened security concerns frequently express reluctance to adopt voice interfaces due to these privacy risks [26].

While our initial VoiceAlign implementation relied on a cloud-based LLM API and inherited these concerns, we subsequently leveraged insights and data from our formative study to fine-tune a small local model. This on-device deployment fundamentally addresses privacy concerns by ensuring all command processing occurs locally without external data transmission, aligning with recommendations for privacy-preserving edge-based AI systems [62]. Beyond privacy, our design incorporates transparency and trust-building mechanisms. VoiceAlign provides real-time visual feedback of detected speech transcription, seeks clarifying questions when commands contain incomplete information, and offers structured command suggestions rather than executing potentially incorrect transformations. In addition, our shim layer functions as an intermediary while preserving the original VUI's features and feedback mechanisms, ensuring that users retain their familiar interaction experience with the underlying system. These features help users develop accurate mental models of system capabilities and limitations, which prior work has identified as essential for appropriate trust calibration in AI-mediated interactions [2, 12].

8.4 Limitations and Future Work

Our work has several limitations. First, using text-to-speech to communicate with the VUI through the virtual channel introduces two challenges: TTS adds latency as commands must be played again for the interface, and VUIs may struggle to recognize robotic

TTS output, occasionally resulting in discarded commands, as we observed during our study. For VUIs supporting direct text input, bypassing TTS could improve reliability.

Second, although our study included participants with varying accents, all were fluent English speakers. Consequently, our findings may not fully reflect the experiences of non-fluent English speakers or users with speech impairments. The cognitive load and task completion strategies of such users when forming voice commands may differ substantially, representing an important direction for future research toward a more comprehensive understanding of VUI command usability.

Third, this work primarily investigates VUIs supporting fixed-format commands. Our analysis of four commercial VUIs and their command structures informed our fully quantified command template, which we reinforced through our formative study. While we believe this template represents command formats typical of VUIs for personal computing devices, it may not encompass the entire VUI design space (e.g., smart home controls, in-car systems, or industrial applications).

Finally, our system maps one voice command to one VUI command, which is a common practice for legacy VUIs. However, with natural commands, it is possible to combine multiple voice commands to allow users more flexibility. Additionally, we focused on text correction commands while VUIs, such as Voice Control, have commands to perform all sorts of different UI control tasks. In the future, we aim to explore such commands to allow users control their devices through natural language commands utilizing the technical capabilities of legacy VUIs underneath.

9 Conclusion

Our work demonstrates how a shimming layer enhances legacy VUI usability by treating them as black boxes. By addressing rigid command formats, restrictive timeouts, and insufficient feedback, our proposed VoiceAlign translates natural voice commands into the precise syntax required by legacy VUI systems. Our evaluation showed substantial improvements: reducing command failures, requiring fewer commands per task, and significantly reducing cognitive and temporal demands. Furthermore, our fine-tuned Gemma 3 270M model enables deployment with over 90% accuracy and 200 ms response time when served locally, eliminating dependence on third-party APIs while enabling real-time interaction on edge devices. This approach offers a practical way to improve voice interactions without replacing existing infrastructure, illustrating how modern LLM techniques can support human-AI interaction.

Acknowledgments

We thank anonymous reviewers for their insightful feedback. This work was supported in part by NSF Grant #2326406. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Science Foundation.

References

[1] Noura Abdi, Xiao Zhan, Kopo M Ramokapane, and Jose Such. 2021. Privacy norms for smart home personal assistants. In *Proceedings of the 2021 CHI conference on human factors in computing systems*. 1–14.

[2] Saleema Amershi, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi Iqbal, Paul N Bennett, Kori Inkpen, et al. 2019. Guidelines for human-AI interaction. In *Proceedings of the 2019 chi conference on human factors in computing systems*. 1–13.

[3] Apple. 2023. Voice Control. https://support.apple.com/en-us/HT210417. [Accessed 13-11-2023].

[4] Noah Apthorpe, Danny Yuxing Huang, Dillon Reisman, Arvind Narayanan, and Nick Feamster. 2018. Keeping the smart home private with smart (er) iot traffic shaping. *arXiv preprint arXiv:1812.00955* (2018).

[5] A. Bryman and R.G. Burgess. 1994. *Analyzing Qualitative Data*. Routledge. https://books.google.com/books?id=KQkotSd9YWkC

[6] Junhwi Choi, Kyungduk Kim, Sungjin Lee, Seokhwan Kim, Donghyeon Lee, Injae Lee, and Gary Geunbae Lee. 2012. Seamless error correction interface for voice word processor. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 4973–4976.

[7] H.H. Clark, H.H. Clark, American Council of Learned Societies, H.C. Clark, and H.H. Clark. 1996. *Using Language*. Cambridge University Press. https://books.google.ie/books?id=DiWBGOP-YnoC

[8] Penny Collings, David Walker, and Michael Wagner. 2002. Usability Evaluation of a Commercial Dictation System. In *Ninth Australian International Conference on Speech Science and Technology*. Australian Speech Science and Technology Associatc, 479–484.

[9] Liwei Dai, Rich Goldman, Andrew Sears, and Jeremy Lozier. 2003. Speech-based cursor control: a study of grid-based solutions. *ACM SIGACCESS Accessibility and Computing* 77-78 (2003), 94–101.

[10] Michael Han Daniel Han and Unsloth team. 2023. *Unsloth*. http://github.com/unslothai/unsloth

[11] ADN Edwards. 2002. Multimodal interaction and people with disabilities. In *Multimodality in language and speech systems*. Springer, 73–92.

[12] Upol Ehsan, Q Vera Liao, Michael Muller, Mark O Riedl, and Justin D Weisz. 2021. Expanding explainability: Towards social transparency in ai systems. In *Proceedings of the 2021 CHI conference on human factors in computing systems*. 1–19.

[13] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. 1995. *Design patterns: elements of reusable object-oriented software*. Pearson Deutschland GmbH.

[14] Debjyoti Ghosh, Pin Sym Foong, Shengdong Zhao, Di Chen, and Morten Fjeld. 2018. EDITalk: towards designing eyes-free interactions for mobile word processing. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–10.

[15] Debjyoti Ghosh, Pin Sym Foong, Shengdong Zhao, Can Liu, Nuwan Janaka, and Vinitha Erusu. 2020. EYEditor: Towards On-the-Go Heads-Up Text Editing Using Voice and Manual Input. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376173

[16] Debjyoti Ghosh, Can Liu, Shengdong Zhao, and Kotaro Hara. 2020. Commanding and re-dictation: Developing eyes-free voice-based interaction for editing dictated text. *ACM Transactions on Computer-Human Interaction (TOCHI)* 27, 4 (2020), 1–31.

[17] Google. 2023. Voice Access. https://support.google.com/accessibility/android/answer/6151848?hl=en. [Accessed 13-11-2023].

[18] Christine A Halverson, Daniel B Horn, Clare-Marie Karat, and John Karat. 1999. The beauty of errors: Patterns of error correction in desktop speech systems.. In *INTERACT*, Vol. 99. 1–8.

[19] Marti A Hearst. 2009. *Search user interfaces*. Cambridge university press.

[20] Nico Herbig, Santanu Pal, Josef van Genabith, and Antonio Krüger. 2019. Multi-modal approaches for post-editing machine translation. In *Proceedings of the 2019 CHI conference on human factors in computing systems*. 1–11.

[21] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2022. Lora: Low-rank adaptation of large language models. *ICLR* 1, 2 (2022), 3.

[22] Rashad Albo Jamara, Nico Herbig, Antonio Krüger, and Josef van Genabith. 2021. Mid-air hand gestures for post-editing of machine translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 6763–6773.

[23] Clare-Marie Karat, Christine Halverson, Daniel Horn, and John Karat. 1999. Patterns of entry and correction in large vocabulary continuous speech recognition systems. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. 568–575.

[24] Nils Krahnstoever, Sanshzar Kettebekov, Mohammed Yeasin, and Rajeev Sharma. 2002. A real-time framework for natural multimodal interaction with large screen displays. In *Proceedings. Fourth IEEE International Conference on Multimodal Interfaces*. IEEE, 349–354.

[25] Per Ola Kristensson and Keith Vertanen. 2011. Asynchronous multimodal text entry using speech and gesture keyboards. In *Twelfth annual conference of the international speech communication association*.

[26] Josephine Lau, Benjamin Zimmerman, and Florian Schaub. 2018. Alexa, are you listening? Privacy perceptions, concerns and privacy-seeking behaviors with smart speakers. *Proceedings of the ACM on Human-Computer Interaction* 2, CSCW (2018), 1–31.

[27] Moxin Li, Wenjie Wang, Fuli Feng, Fengbin Zhu, Qifan Wang, and Tat-Seng Chua. 2024. Think twice before assure: Confidence estimation for large language models through reflection on multiple answers. *arXiv preprint arXiv:2403.09972* (2024).

[28] Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*. 74–81.

[29] Susan Lin, Jeremy Warner, JD Zamfirescu-Pereira, Matthew G Lee, Sauhard Jain, Shanqing Cai, Piyawat Lertvittayakumjorn, Michael Xuelin Huang, Shumin Zhai, Björn Hartmann, et al. 2024. Rambler: Supporting Writing With Speech via LLM-Assisted Gist Manipulation. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. 1–19.

[30] Can Liu, Siying Hu, Li Feng, and Mingming Fan. 2022. Typist Experiment: an Investigation of Human-to-Human Dictation via Role-play to Inform Voice-based Text Authoring. *Proceedings of the ACM on Human-Computer Interaction* 6, CSCW2 (2022), 1–33.

[31] Xingkun Liu, Arash Eshghi, Pawel Swietojanski, and Verena Rieser. 2021. Benchmarking natural language understanding services for building conversational agents. In *Increasing naturalness and flexibility in spoken dialogue interaction: 10th international workshop on spoken dialogue systems*. Springer, 165–183.

[32] Nathan Malkin, Joe Deatrick, Allen Tong, Primal Wijesekera, Serge Egelman, and David Wagner. 2019. Privacy attitudes of smart speaker users. *Proceedings on Privacy Enhancing Technologies* 2019, 4 (2019).

[33] Arthur E McNair and Alex Waibel. 1994. Improving recognizer acceptance through robust, natural speech repair. In *3rd International Conference on Spoken Language Processing (ICSLP 1994)*. ISCA.

[34] Microsoft. 2023. Voice Access. https://support.microsoft.com/en-us/topic/get-started-with-voice-access-bd2aa2dc-46c2-486c-93ae-3d75f7d053a4. [Accessed 13-11-2023].

[35] Nuance. 2023. Dragon Speech Recognition. https://www.nuance.com/dragon.html. [Accessed 13-11-2023].

[36] OpenAI. 2025. ChatGPT on your desktop. https://openai.com/chatgpt/desktop/ Accessed: 2025-04-10.

[37] Sharon Oviatt. 1997. Mulitmodal interactive maps: Designing for human performance. *Human–Computer Interaction* 12, 1-2 (1997), 93–129.

[38] Sharon Oviatt. 2000. Taming recognition errors with a multimodal interface. *Commun. ACM* 43, 9 (2000), 45–51.

[39] Sharon Oviatt and Philip Cohen. 2000. Perceptual user interfaces: multimodal interfaces that process what comes naturally. *Commun. ACM* 43, 3 (2000), 45–53.

[40] Sharon Oviatt, Phil Cohen, Lizhong Wu, Lisbeth Duncan, Bernhard Suhm, Josh Bers, Thomas Holzman, Terry Winograd, James Landay, Jim Larson, et al. 2000. Designing the user interface for multimodal speech and pen-based gesture applications: State-of-the-art systems and future research directions. *Human-computer interaction* 15, 4 (2000), 263–322.

[41] Sharon Oviatt and Philip R Cohen. 2015. The Paradigm Shift to Multimodality in Contemporary Computer Interfaces. Morgan & Claypool Publishers.

[42] Sharon Oviatt and Robert VanGent. 1996. Error resolution during multimodal human-computer interaction. In *Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP'96*, Vol. 1. IEEE, 204–207.

[43] Kseniia Palin, Anna Maria Feit, Sunjun Kim, Per Ola Kristensson, and Antti Oulasvirta. 2019. How do people type on mobile devices? Observations from a study with 37,000 volunteers. In *Proceedings of the 21st International Conference on Human-Computer Interaction with Mobile Devices and Services*. 1–12.

[44] Aung Pyae and Tapani N Joelsson. 2018. Investigating the usability and user experiences of voice user interface: a case of Google home smart speaker. In *Proceedings of the 20th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct*. 127–131.

[45] Ashwin Ram, Rohit Prasad, Chandra Khatri, Anu Venkatesh, Raefer Gabriel, Qing Liu, Jeff Nunn, Behnam Hedayatnia, Ming Cheng, Ashish Nagar, et al. 2018. Conversational ai: The science behind the alexa prize. *arXiv preprint arXiv:1801.03604* (2018).

[46] Anthropic Research. 2024. *Introducing the Model Context Protocol.* Technical Report. Anthropic. https://www.anthropic.com/news/model-context-protocol

[47] Nathaniel Robinson, Perez Ogayo, David R Mortensen, and Graham Neubig. 2023. ChatGPT MT: Competitive for high-(but not low-) resource languages. In *Proceedings of the Eighth Conference on Machine Translation*. 392–418.

[48] Andrew Sears, Jinhuan Feng, Kwesi Oseitutu, and Claire-Marie Karat. 2003. Hands-free, speech-based navigation during dictation: difficulties, consequences, and solutions. *Human-computer interaction* 18, 3 (2003), 229–257.

[49] Korok Sengupta, Sabin Bhattarai, Sayan Sarcar, I Scott MacKenzie, and Steffen Staab. 2020. Leveraging error correction in voice-based text entry by Talk-and-Gaze. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–11.

[50] Khe Chai Sim. 2010. Haptic voice recognition: Augmenting speech modality with touch events for efficient speech recognition. In *2010 IEEE spoken language technology workshop*. IEEE, 73–78.

[51] Khe Chai Sim. 2012. Speak-as-you-swipe (SAYS) a multimodal interface combining speech and gesture keyboard synchronously for continuous mobile text entry. In *Proceedings of the 14th ACM international conference on Multimodal interaction*. 555–560.

[52] Shyamli Sindhwani, Christof Lutteroth, and Gerald Weber. 2019. Retype: Quick text editing with keyboard and gaze. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–13.

[53] SnapLogic. 2023. Legacy tech upgrades cost the average business nearly $3M last year. https://www.ciodive.com/news/legacy-technology-technical-debt-costs-enterprise-data-AI/721885/ Accessed: 2025-04-10.

[54] Matthias Sperber, Graham Neubig, Christian Fügen, Satoshi Nakamura, and Alex Waibel. 2013. Efficient speech transcription through respeaking.. In *Interspeech*. 1087–1091.

[55] OrCam Staff. [n. d.]. How Voice-Activated Technology Improves the Lives of Blind People - OrCam — orcam.com. https://www.orcam.com/en-us/blog/how-voice-activated-technology-improves-the-lives-of-blind-people. [Accessed 03-04-2024].

[56] Bernhard Suhm, Brad Myers, and Alex Waibel. 2001. Multimodal error correction for speech user interfaces. *ACM transactions on computer-human interaction (TOCHI)* 8, 1 (2001), 60–98.

[57] John Sweller. 2011. Cognitive load theory. In *Psychology of learning and motivation*. Vol. 55. Elsevier, 37–76.

[58] Andries Van Dam. 1997. Post-WIMP user interfaces. *Commun. ACM* 40, 2 (1997), 63–67.

[59] Keith Vertanen and Per Ola Kristensson. 2009. Automatic selection of recognition errors by respeaking the intended text. In *2009 IEEE Workshop on Automatic Speech Recognition & Understanding*. IEEE, 130–135.

[60] Keith Vertanen, Kyle Montague, Mark Dunlop, Ahmed Sabbir Arif, Xiaojun Bi, and Shiri Azenkot. 2017. Ubiquitous text interaction. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. 566–573.

[61] DeLiang Wang and Jitong Chen. 2018. Supervised speech separation based on deep learning: An overview. *IEEE/ACM transactions on audio, speech, and language processing* 26, 10 (2018), 1702–1726.

[62] Dianlei Xu, Tong Li, Yong Li, Xiang Su, Sasu Tarkoma, Tao Jiang, Jon Crowcroft, and Pan Hui. 2021. Edge intelligence: Empowering intelligence to the edge of network. *Proc. IEEE* 109, 11 (2021), 1778–1837.

[63] Yong Xu, Jun Du, Li-Rong Dai, and Chin-Hui Lee. 2014. A regression approach to speech enhancement based on deep neural networks. *IEEE/ACM transactions on audio, speech, and language processing* 23, 1 (2014), 7–19.

[64] Lishuang Zhan, Tianyang Xiong, Hongwei Zhang, Shihui Guo, Xiaowei Chen, Jiangtao Gong, Juncong Lin, and Yipeng Qin. 2024. TouchEditor: Interaction design and evaluation of a flexible touchpad for text editing of head-mounted displays in speech-unfriendly environments. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 7, 4 (2024), 1–29.

[65] Biao Zhang, Barry Haddow, and Alexandra Birch. 2023. Prompting large language model for machine translation: A case study. In *International Conference on Machine Learning*. PMLR, 41092–41110.

[66] Maozheng Zhao, Wenzhe Cui, IV Ramakrishnan, Shumin Zhai, and Xiaojun Bi. 2021. Voice and touch based error-tolerant multimodal text editing and correction for smartphones. In *The 34th Annual ACM Symposium on User Interface Software and Technology*. 162–178.

[67] Maozheng Zhao, Henry Huang, Zhi Li, Rui Liu, Wenzhe Cui, Kajal Toshniwal, Ananya Goel, Andrew Wang, Xia Zhao, Sina Rashidian, Furqan Baig, Khiem Phi, Shumin Zhai, IV Ramakrishnan, Fusheng Wang, and Xiaojun Bi. 2022. EyeSayCorrect: Eye Gaze and Voice Based Hands-Free Text Correction for Mobile Devices. In *27th International Conference on Intelligent User Interfaces* (Helsinki, Finland) *(IUI '22)*. Association for Computing Machinery, New York, NY, USA, 470–482. https://doi.org/10.1145/3490099.3511103

[68] Shengdong Zhao, Felicia Tan, and Katherine Fennedy. 2023. Heads-Up Computing Moving Beyond the Device-Centered Paradigm. *Commun. ACM* 66, 9 (2023), 56–63.