8-2020

Commanding and re-dictation: Developing eyes-free voice-based interaction for editing dictated text

Debjyoti GHOSH
National University of Singapore

Can LIU
City University of Hong Kong

Shengdong ZHAO
National University of Singapore

Kotaro HARA
Singapore Management University, kotarohara@smu.edu.sg

Citation

GHOSH, Debjyoti; LIU, Can; ZHAO, Shengdong; and HARA, Kotaro. Commanding and re-dictation: Developing eyes-free voice-based interaction for editing dictated text. (2020). *ACM Transactions on Computer-Human Interaction*. 27, (4), 1-31. Research Collection School Of Computing and Information Systems.

Available at: https://ink.library.smu.edu.sg/sis_research/5889

Commanding and Re-Dictation: Developing Eyes-Free Voice-Based Interaction for Editing Dictated Text

DEBJYOTI GHOSH, NUS Graduate School for Integrative Sciences and Engineering, National University of Singapore
CAN LIU, School of Creative Media, City University of Hong Kong
SHENGDONG ZHAO, School of Computing, National University of Singapore
KOTARO HARA, School of Information Systems, Singapore Management University

Existing voice-based interfaces have limited support for text editing, especially when *seeing* the text is difficult, e.g., while walking or cooking. This research develops voice interaction techniques for eyes-free text editing. First, with a Wizard-of-Oz study, we identified two primary user strategies: using commands, e.g., "REPLACE *go* with *goes*" and re-dictating over an erroneous portion, e.g., correcting "he go there" by saying "he goes there." To support these user strategies with an actual system implementation, we developed two eyes-free voice interaction techniques, *Commanding* and *Re-dictation*, and evaluated them with a controlled experiment. Results showed that while Re-dictation performs significantly better for more semantically complex edits, Commanding is more suitable for making one-word edits, especially deletions. We developed *VoiceRev* to combine both the techniques in the same interface and evaluated it with realistic tasks. Results showed improved usability of the combined techniques over either of the two techniques used individually.

CCS Concepts: • **Human-centered computing → Interaction techniques**; **Text input**;

Additional Key Words and Phrases: Text editing, commanding, re-dictation, eyes-free, voice-based text editing, voice interaction, voice user interfaces

ACM Reference format:
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 Trans. Comput.-Hum. Interact.* 27, 4, Article 28 (August 2020), 31 pages.
https://doi.org/10.1145/3390889

1 INTRODUCTION

In Human–Computer Interaction (HCI), text editing typically involves visual engagement: Although text *input* can be performed using various techniques, e.g., physical keyboard, touch-based

This work was supported in part by the NUS Advanced Robotics Centre.

Authors' addresses: D. Ghosh, NUS Graduate School for Integrative Sciences and Engineering, National University of Singapore, Singapore 119077; email: debjyoti@u.nus.edu; C. Liu, School of Creative Media, City University of Hong Kong, Hong Kong; email: canliu@cityu.edu.hk; S. Zhao, School of Computing, National University of Singapore, Singapore 119077; email: zhaosd@comp.nus.edu.sg; K. Hara, School of Information Systems, Singapore Management University, Singapore 178902; email: kotarohara@smu.edu.sg.

1073-0516/2020/08-ART28 $15.00
https://doi.org/10.1145/3390889

interactions, or by using voice-input, the *feedback* (output) and *outcome evaluation* [21] are done through on-screen displays and, hence, require visual engagement.

Voice-only interfaces, such as Amazon's Echo, Google Home, and Xiaomi's XiaoAI are now becoming mainstream [1]. These interfaces allow users to perform tasks with computing devices eyes-free and hands-free, which is especially useful when the user is engaged in other tasks, on the go or in the home environment. The emergence of such interfaces paints a desirable future scenario of flexible interaction with computers: the ability to interact with computing devices without the need to hold or be in contact with the device, so that the whole body is free to engage in other tasks.

However, the capabilities of such interfaces are still fairly limited—while consumer-grade voice assistants like Google Assistant and Alexa already enable people to perform simple tasks (e.g., check the weather, search directions, and set a reminder), their efficacy to support users to perform more complex tasks such as editing text is underexplored, especially when visual feedback is precluded. Although services like Alexa's *Mail Box* "skill" allow users to compose e-mails or short text messages via speech in an eyes-free manner, user reviews suggest that the interactions are severely limiting and pose major usability challenges [2].

A major challenge for speech-based interaction with text is the difficulty in *editing* the dictated text [3]. While the increasingly mature speech-to-text technology can automatically transcribe user utterances, the dictated text often needs to be corrected because of errors like speech-recognition errors and incoherence of the author's thoughts during dictation. The challenge of speech-based text editing is exacerbated in eyes-free situations. Existing tools like *Mail Box* can read out the text that the user wrote via speech, allowing them to "proofread" the composed text without seeing it. However, the linear [16, 27, 33] and temporal [16, 37] nature of audio makes the audio-based proofreading prohibitively slow and error-prone. Ghosh et al. [11] showed that without visual feedback users failed to complete editing tasks most of the time, and even when they could, the accuracy levels were low. Yet, as evident in both user research [11] and in consumer-grade products like *Mail Box*, it is desirable to allow users to conveniently compose and edit short e-mails, messages, or social media posts in an eyes-free fashion, using voice input, especially when they are on the go or being occupied by other tasks such as cooking.

Previous work [11] has established the feasibility and desirability of eyes-free text editing. However, the interactions were designed as per the designer's choice and, thus, may not support the users' most preferred speaking strategies. Although such interactions are suitable for early prototypes, they are not useful in determining what interactions users would naturally choose to edit text eyes-free *without* regard for recognition or technical errors. We began by exploring natural verbal interactions users do when speaking to a system for composing and editing text eyes-free. To avoid the influence of current technical limitations, we conducted a formative Wizard-of-Oz study with 10 participants. One experimenter acted as a computer to simulate a smart agent (wizard) that would help the users to compose and subsequently edit the composed text. We observed that the participants commonly used two editing strategies throughout the study: (1) editing by using commands that did not adhere to any strict syntax, e.g., "please CHANGE *this* to *that*"; and (2) editing by re-speaking over the erroneous part of the text with the correct phrasing, e.g., editing "he go there" by simply speaking "he goes there."

Techniques that support these two identified strategies have been explored in previous research [19, 32, 34, 35], but not in the eyes-free context. Unlike text editing with visual feedback, where the user can see the erroneous text prior to correcting it, eyes-free text editing needs user to precisely remember the dictated text (from an audio feedback of the text), in order to both identify and correct an erroneous portion. In that, eyes-free text editing needs to support interactions to minimize the user's cognitive load of recall, support barge-in correction utterances, and provide real-time response for *outcome evaluation* [11]. Hence, additional investigation is necessary to re-study the existing techniques so that they support the identified strategies in the eyes-free context. Also, there is a lack of understanding on how the two strategies compare with each other in influencing the users' text-editing performance. Therefore, in this work, we investigate the following questions. (1) How can we adapt the existing techniques to the context of eyes-free? (2) How does each technique affect the user performance of editing text? (3) Given that existing text editing interfaces do not support both techniques together, is it necessary and sufficient to support either technique individually for eyes-free use?

First, we designed a voice-based interface and implemented two eyes-free techniques, *Commanding* and *Re-dictation*, to support the two main identified user strategies of editing from the Wizard-of-Oz study. To investigate the usability of these two techniques, we evaluated their efficiency and ease of use with a controlled experiment with 16 participants. The tasks for this experiment tested the performance of each technique in editing text of different complexities defined through carefully chosen parameters. Results showed that each technique had its own advantage in eyes-free text editing: For making simple one-word edits (e.g., insert/delete a single word), Commanding was either faster (deletion) or performed similarly to Re-dictation (insertion), whereas for making longer and more complex corrections, Re-dictation was both the faster and the preferred technique. Also, the results suggested that combining both the techniques in a coherent interface might improve the usability of eyes-free text-editing systems as it would allow users a preferred choice of technique based on the edit length and complexity of the intended correction.

We implemented a unified system, VoiceRev, to support both the techniques together and evaluated it with a realistic study involving eight participants. Results confirmed that our findings from the controlled study also hold with realistic tasks. Across 266 successful revision trials, 62% were correction by Re-dictation and almost 29% were command-based *deletions*. Also, participants used Re-dictation to correct longer regions of incorrect text, whereas the average number of words corrected with Commanding was in the range of one to three. Finally, we discuss the implications of our findings for the design of future voice-based systems.

Our contributions include (1) an understanding of naturally emerging user behavior for eyes-free text editing using speech; (2) design and implementation of a voice interface, VoiceRev, that supports two text-editing techniques: *Commanding* and *Re-dictation* together and can be used eyes-free; (3) a controlled experiment providing an in-depth understanding of the two techniques; and (4) a study with realistic tasks to evaluate the VoiceRev system and confirm our findings from the controlled study.

2 RELATED WORK

There are three broad areas our work relates to: use of voice input as an eyes-free input modality, use of multimodality in text editing, and one-step text editing with voice input.

2.1 Voice for Eyes-Free Input

Voice as a form of eyes-free *output* (e.g., [9, 25, 30, 31, 39]) has been extensively studied. In contrast, not many studies in the HCI literature have explored voice as a form of eyes-free *input*. *JustSpeak* [40] explored eyes-free voice input as a means to facilitate device-wide voice control of an Android device. This work was targeted towards blind and motion-impaired users. Azenkot et al. studied voice as an input modality for nonvisual text entry in mobile devices [3]. Findings of this work revealed that though speech input was an efficient alternative to using on-screen keyboards for text entry, users found it difficult to correct misrecognized text from the speech recognizer's output and had to fall back on using the on-screen keyboard for the correction.

Voice-based text correction has been explored in voice dialing systems [10]. These systems allowed the user to speak lengthy telephone numbers (say, 10–12 digits) in chunks of multidigit strings of variable length. The user would pause after every chunk and wait for the system to read back the recognized digits, thereby verifying if the recognition was correct. If incorrect, the user could issue commands such as "CLEAR" to discard the dictated chunk and say it again. However, such voice dialing systems were limited in their text (digit) entry speed and did not allow users to navigate through the text. Similar to voice dialing systems, voice dictation has been explored in automotive (in-car) dictation editors [7]. Like with voice dialing, users voice dictate the text by speaking one short segment at a time and wait for the system to play back an audio of the recognized segment for verification. Additionally, users can navigate through the segments using a steering wheel interface and choose to delete and re-speak segments containing errors in them. However, these systems are targeted towards a specific use case, i.e., entering and editing text while driving, and, like voice dialing systems, are limited in their text entry speeds and editing capabilities. Hence, both voice dialing and in-car dictation systems are impractical for entering/editing longer texts. Furthermore, these systems restrict the users' "customary, natural" speaking behavior [6].

Eyes-free use of voice input has recently been explored in the context of text editing by Ghosh et al.'s *EDITalk* [11]. This research showed the challenges of using existing voice-based dictation software like Dragon Naturally Speaking for end-to-end eyes-free operation. As a solution, the authors proposed EDITalk, an eyes-free command-based text-editing system that can be operated with voice-only input and output. However, this system explored eyes-free voice-based text editing with command-only interaction, which might not be reflective of the users' preferred mode of interaction. Also, command-based editing might not be optimal in the eyes-free context as using commands requires the user to memorize the exact command syntax and speak it out precisely. This recall of information might add on to the heavy cognitive load of eyes-free listening and editing [11].

2.2 Multimodal Text Editing

Although not within the eyes-free context, there exists a body of literature exploring the *visual* use of commanding and re-dictation for voice-based text editing. Halverson et al. [13] had studied user patterns of voice-based error correction in desktop speech systems, and found out that using a single modality of input (re-dictation) might lead to spiral depths [24] and cascades [13], which slows down the error-correction process. They suggested that switching modalities of input from voice-only re-dictation to voice+mouse or voice+keyboard is helpful to cut down on the error-correction time. Since then, several other works have explored multimodality in the context of error correction. Suhm et al. [32] had explored the use of speech along with mouse, keyboard, and stylus and found that using multimodal input increases the error-correction speed.

Furthermore, Oviatt [23] had previously suggested that combining speech input with additional input modalities might improve the speech-recognition accuracy. Later, researchers showed that the accuracy of text entry can be improved by combining speech input with input from a gestural keyboard, both asynchronously [15] and synchronously [28].

Different from this body of work, we focus on eyes-free voice-based editing without the help from keyboard, mouse, or visual display. However, we draw inspiration from the previous research, which suggests that instead of sticking to one particular strategy, a combination of multiple strategies can lead to an improvement in the performance and usability of error correction.

2.3 One-Step Voice-Based Text Editing

The re-dictation technique has been explored in the visual-text-editing context, under two approaches: two step versus one step. The two-step re-dictation approach was presented in an earlier work [13], where the erroneous text is first selected with a select instruction (target-based navigation) and then the correction is dictated to replace the selected text (correction). As the requirement of selecting the text first is an additional step that requires users' time and effort, McNair and Weibel proposed to simplify the two-step approach into a one-step approach [19]. The one-step approach allowed users to correct erroneous text by re-speaking over the erroneous part using the correct wordings, e.g., correcting "he go there" by saying "he goes there."

To achieve a one-step correction, the text-editing system needs to understand which part of the text the user intends to replace; this can be a challenging task. Several previous studies have attempted to solve this problem. In these works, although a diverse terminology has been used to describe the correction technique: "fluid" text correction [36], "seamless error correction" [5], and correction through "re-speaking" [29, 34], the terms essentially embody the same one-step approach.

One-step correction involves first identifying the error region and then correcting it. One attempt at identifying the error region has been using automatic alignment models [34]. Choi et al. [5] suggested a model for both identification and subsequent automatic correction. Vertanen and Kristensson [35] further proposed a merge model to improve the recognition accuracy of the correction utterance.

While the previous works on re-dictation under the visual-text-editing context were relevant and extremely valuable, our work differs from them in two aspects. (1) Instead of applying re-dictation with a visual display, we focus on its use in eyes-free scenarios. As pointed out in the Introduction, unlike voice input with visual feedback, eyes-free voice interaction (voice as both the input and output modalities) needs to support barge-in interactions, minimize users' cognitive load of recall, and provide real-time audio feedback of the interaction outcome. (2) Instead of focusing on improving the performance of re-dictation, we aim at understanding when and how such techniques should be used for eyes-free text editing with voice, and how correction by re-dictation can complement the use of commands to achieve a better user experience.

3 STUDY 1: WIZARD-OF-OZ STUDY

We started our investigation with a Wizard-of-Oz study in order to understand how users naturally perform eyes-free dictation and editing tasks when not constrained by current technical limitations. An experimenter simulated a system that allowed users to speak freely for composing and revising text. Participants were not aware of the system being operated by a human experimenter.

3.1 Study Design

3.1.1 Participants. We recruited 10 participants (5 female, 5 male; mean age = 25.2 years, SD = 4.05), of whom 4 had never used a dictation application before, while the other 6 used one occasionally. As an exploratory study, we intentionally recruited people with different levels of exposure to voice-based interfaces to observe if there are any differences in how they interact with the devices.

3.1.2 Apparatus. Participants performed the task eyes-free, seated by speaking into a Blue Yeti condenser microphone, while the system was run by an experimenter seated behind a large piece of cardboard that separated the experimenter from the participant. The experimenter operated an in-house designed web-based tool running on a 13" MacBook Pro (2017 edition) laptop. The web tool was developed using JavaScript and combined a Text-to-Speech (TTS) reader for audio playback with Google's Speech-to-Text Application Programming Interface (API) for speech transcription. This tool transcribed the participants' speech into text and allowed the experimenter to edit it and play back an audio of the text to the participant. This procedure allowed us to bypass the technical

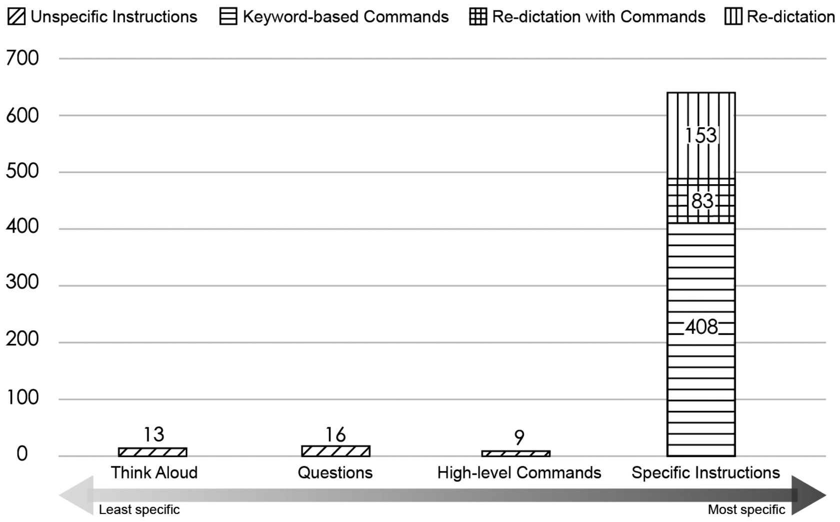

Fig. 1. Categories of text-editing instructions used by participants in the Wizard-of-Oz setting.

limitations of existing techniques in dealing with speech repair and editing dictated text [19, 22], and allowed the user to speak freely and naturally. In addition, the experimenter occasionally played back a minimal system response, such as “I did not understand your instruction” or “I could not find the words you mentioned,” as necessary to aid the interaction.

3.1.3 Task and Procedure. Participants started with a training task of composing a message to a family member or a friend. After getting familiar with the system, the participants were asked to perform four tasks. Each task required the participants to compose and revise a piece of text about 150 words in length. For two of the tasks, the topic was informal social media posts, while for the other two the topic was formal professional e-mails. As part of the task-completion criteria, the participants were asked to make sure that the content was ready to be published on a social media platform or sent out to an actual person. The participants were instructed to speak freely and not be limited by any prior experience with a voice user interface. While revising the content, at any point during the task, participants could instruct the system to read back the text. By default, anytime a participant made a correction utterance, the experimenter (Wizard) would manually modify the text as instructed by the participant and play back the modified text from just before the modified portion. After the tasks had finished, participants were interviewed for about 15 minutes to know their subjective preferences and strategies of performing the text-modification operations. The entire experiment lasted for approximately an hour.

3.1.4 Data Collection and Analysis Method. We recorded the audio of the speech interaction and the screen of the experimenter. Also, all interviews were audio recorded. A member of the research team transcribed the recorded audio files for subsequent analysis. We used thematic analysis to identify recurring themes in the data through open and axial coding.

3.2 Findings

As mentioned before, thematic analysis was performed on the transcripts of the speech interaction. We analyzed how participants formed their instructions to the system. As Figure 1 shows, a spectrum of instructing behavior emerged from the data, with the instructions ranging from being very unspecific to very specific.

We coded six categories from the utterances used for delivering the edit instructions: *think aloud*, *questions*, *high-level commands*, *low-level commands*, *re-dictation*, and *re-dictation with commands*. *Think aloud* includes utterances that express a participant's state of mind or thought process during the interaction and often convey some intention, e.g., "I don't remember what I was supposed to say." *Questions* are utterances where the participants had asked a question to the system, e.g., "Did I say 'xxx'?" *High-level commands* are commands that are abstract and require additional processing to become concrete editing operations, such as "There should be only one 'whether' in the sentence." (Here, the concrete edit operation implied was to delete one of the two "whether" that were placed right next to each other in the text.) Unlike high-level commands, *low-level commands* are *specific command-based* instructions to edit the text, e.g., "DELETE the phrase 'had been'." For the low-level commands, participants used keywords like DELETE or CHANGE to specify concrete editing operations (intents). Utterances categorized as *re-dictation* are the ones in which participants re-dictated an erroneous portion of the text with the re-dictated utterance containing the desired change(s). For example, "I will waiting for" was corrected by saying "will be waiting."

Finally, the *re-dictation with commands* category includes utterances where participants combined the use of keywords with re-dictation, e.g., "Can you please CHANGE to 'had been'?" (Here, the intended correction was to change from "have been" to "had been.") The correction utterance in the previous example requests a change operation without specifying which portion of the text the change should apply to. Also, the use of the keyword is redundant as the change intent is evident even without the keyword. Hence, the utterance can in effect be interpreted as a re-dictation utterance. On the other hand, some utterances in this category specify the same instruction using both a command and re-dictation simultaneously, e.g, "CHANGE the 'for' to 'to' and make it 'we went to the movie'." (The intended correction was to change from "we went for the movie" to "we went to the movie.") In this example, the utterance can be decomposed into two complete instructions—one using low-level commands and the other using re-dictation. Therefore, although we coded a separate category for *re-dictation with commands*, in effect they can be expressed using either low-level commands or re-dictation.

The first three categories—think aloud, questions, and high-level commands—were further combined into a higher order category: *unspecific instructions*, owing to the vagueness in their expression. The process of inferring concrete edit operations from the unspecific instructions requires deep contextual understanding and the ability to interpret abstract thinking. In contrast, the last three categories—low-level commands, re-dictation, and re-dictation with commands—were further combined into a higher order category: *specific instructions*, owing to the concreteness in their expression.

3.2.1 Interaction Strategies for Editing. Figure 1 illustrates the occurrence of different instruction types and places them on a spectrum of specificity. Analysis of all the identified instruction types and editing strategies revealed two concrete strategies that were used the most—use of *re-dictation* and *low-level, keyword-based commands*. Additionally, participants switched between text composition and revision in two different styles. Seven participants first composed the entire content and then revised it from the beginning, while three participants alternated multiple times between composing and revising (i.e., composed a few sentences each time and then revised it before moving on to compose the next set of sentences).

The findings of this study warrant a deeper investigation into the two major user strategies of editing text eyes-free. While the other identified strategies are low in their number of occurrences, they also lie beyond the scope of what can be achieved with the state-of-the-art in computing and machine intelligence. Also, as observed in this study, there was a minimal overlap between the composition and revision phases. The common user behavior was to start the revision phase only after the composition phase had ended. Therefore, we do not study the users' switching behavior between the two phases in the rest of this article. Hence, the scope of our subsequent explorations is to study command-based and re-dictation-based instructions for editing text eyes-free in the revision phase. In that, our aim is to understand the opportunities and challenges presented by each strategy under different task constraints.

4 DEVELOPING AN EYES-FREE VOICE INTERFACE FOR TEXT EDITING

Based on our findings from Study 1, we developed a voice interface for eyes-free text editing that supports two interaction techniques: *Commanding* and *Re-dictation*. Commanding supports command-based editing without having to adhere to a strict syntax, e.g., "CHANGE *this* to *that*," while Re-dictation supports correction by re-speaking over the erroneous portions of the text with the correct phrasing, e.g., in the phrase "a hundred splendid sons," the "*hundred*" can be changed to "*thousand*" by saying "thousand splendid sons," "a thousand," or any other similar phrase that repeats over an existing part of the text and contains the desired change. The system would then automatically compute the portion of the original text that needs to be modified to achieve the desired correction, based on the similarity between the re-spoken text and the original text.

Existing techniques that support the two strategies—command-based correction and correction by re-dictation—rely on visual feedback for the editing process. The aim of our implementation was to adapt these existing techniques for eyes-free use. For the rest of this article, we call the eyes-free adaptation of the two techniques as Commanding and Re-dictation, respectively.

This section describes the implementation of both the Commanding and the Re-dictation techniques. First, we discuss the components that are common to both the techniques followed by the specific implementation details of each technique individually. Note that, for all examples in this section, we use the famous English pangram: "The quick brown fox jumps over the lazy dog." In our examples, any deviation from the pangram's original form, e.g., "*jumped* over" or "The quick *fox*..." would denote an error in the sentence with the error italicized and would be corrected back into its original form.

4.1 Common System Architecture

Figure 2 shows the system architecture that serves as the base framework on which specific implementations for the Commanding and Re-dictation techniques were developed. The system consists of a speech-to-text (speech-recognition) engine for transcribing user-dictated text and voice-based text-editing instructions (correction utterances), a text-to-speech (TTS) engine for audio playback of the transcribed text, a language parser for parsing the users' correction utterances, an instruction processing subsystem to compute the text update parameters and perform the required system actions based on the user instructions, and a text editor to preserve the dictated text for postediting. For speech recognition, we used Google's state-of-the-art Cloud Speech-to-Text API.[1] For TTS, *talkify.js*, which is a JavaScript-based TTS library, was used. Another JavaScript-based library, *quill.js*, was used to implement the text editor. *Quill.js* allows modification of the text programmatically with a JavaScript API. The rest of the components were designed in-house.

The system allows the user to voice dictate a piece of text, which gets transcribed and recorded in the text editor. Google's speech-to-text library automatically removes filler words such as um, uh, er, and so on, thereby precluding the need for additional preprocessing of the user utterance for the removal of filler words. When the user opts to revise the text, the TTS engine provides an audio playback of the transcribed text to the user. By default, the *talkify* library reads out punctuation marks in the text by appropriate pauses and inflections in the TTS voice. During the audio

[1] https://cloud.google.com/speech-to-text.

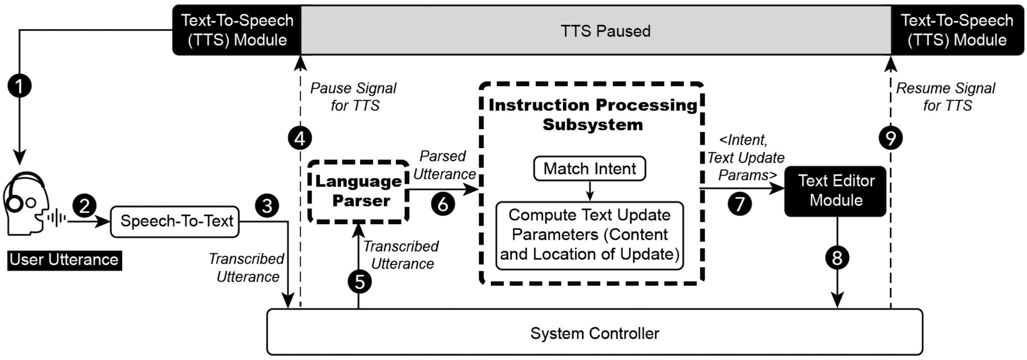

Fig. 2. Common system architecture—components and input/output. Numbers 1–9 show the order of events and the information flow between the system components in the eyes-free text-revision process. (1) User listens to an audio of the dictated text. (2) User makes a correction utterance. (3) The utterance is transcribed to text. (4) System controller signals TTS to stop reading the text. (5) The transcribed utterance is forwarded to the language parser for parsing. (6) The parsed utterance and the text stored in the text editor (not shown) serves as input to the instruction processing subsystem; (7) The instruction processing subsystem after extracting the correction intent from the utterance and computing the text update parameters sends the intent and the update parameters to the Text Editor module. (8) The text editor module updates the text in the text editor and sends the control back to the system controller. (9) The system controller signals the TTS to resume reading; the TTS picks up the updated text from the text editor and resumes reading (not shown).

Table 1. List of Supported Intents

Intent	Parameters
Insert	content, location
Delete	content, location (optional)
Change	old_content, new_content, location (optional)
Read	location (optional), read_mode (optional)
Repeat	location (optional), repeat_mode (optional)
Undo/Redo	nonparameterized
Stop/Resume	nonparameterized
Get_Location	nonparameterized

playback, a user can edit the text by speaking a voice instruction. As the user starts speaking, the TTS stops the playback and the text modification (as instructed by the user) happens in real-time. Finally, the TTS resumes reading from just before[2] the modified portion of the text and continues reading until the user interrupts again or the whole text has been read.

The system supports two sets of operations: one set for common administrative operations and the other set for *core* text-modification operations [26]. The common administrative operations include *repeat*, *undo*, *redo*, *stop*, *resume*, and *get_location* (see Table 1). This set of operations is supported by both the text-modification techniques. The repeat operation repeats the last sentence

[2] The resumption happens 25 character positions before (which is about 5 words before) the position of modification. This additional context information helps to maintain a continuity in the interaction despite the interruption. Computing 25 character positions before the position of modification might result in a position that is in the previous sentence or in the middle of a word. The former scenario is resolved by resuming the reading from the start of the modified sentence, while the latter is resolved by resuming the reading from the start of the bounding word.

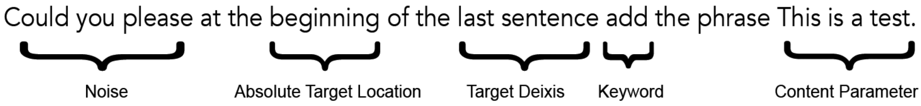
Could you please at the beginning of the last sentence add the phrase This is a test.
Noise
Absolute Target Location
Target Deixis
Keyword
Content Parameter

Fig. 3. A correction utterance labeled in Dialogflow.

read by the TTS, the *undo* and *redo* operations undo or redo the last operation, the *stop* operation stops the TTS, and the *resume* operation resumes the TTS. The *get_location* operation allows the user to inquire the sentence number of the current sentence being read by the TTS. The core text-modification operations are responsible for modifying the text and are handled differently by the two techniques.

4.2 Commanding Technique

Our results from Study 1 showed that users often use different keywords for commands with the same intent. Also, we noticed that when speaking out the command, users do not follow a strict syntax. To accommodate such behaviors, we introduced two important design considerations into the design and implementation of our command-based modification technique.

First, consistent with the user behavior, the system should accommodate words/phrases that are synonymous with (1) keywords for specifying the intent, and (2) words that represent deictic references, e.g., ‘this’, ‘after’, and so on. We designed our system to recognize several keywords for each intent. For example, all the following keyword-preposition pairs: **CHANGE** to, **SWITCH** to, **REPLACE** by, **REPLACE** with, and **TRANSFORM** to, represent the *Change* intent. The keywords were informed by user utterances from Study 1. We used Google’s *Dialogflow*[3] to label the keywords and categorize them based on intent. Also, our system supports synonyms in identifying deictic references. For example, in a sentence with two occurrences of the word “*fox*,” the instruction “**DELETE** *fox*” after the second “*fox*” has been read out (by the TTS) would yield the same result as saying “**SCRATCH** the second *fox* out” at any point during the sentence playback.

The previous example also clarifies that the conflict between repeated occurrences of the target word is resolved in favor of the most recently read one. We based this design decision on our assumption that since speech is linear and temporal in nature, users are more likely to have referred to a piece of information that is more recent than a previous occurrence of the same information.

Second, the system should accommodate commonly occurring “conversational-style” elements in the user utterance and parse them out as noise. For example, if the user commands an insert operation with an utterance such as: “Could you please **INSERT** the phrase *jumped over* **after** *quick brown fox* in this sentence?” then the system should parse out the verbiage and interpret the intent as “**INSERT** *jumped over* **after** *quick brown fox*.” Luger and Sellen [18] had noted the presence of these “conversational-style” elements in users’ conversations with their conversational agents.

We used *Dialogflow* to train a hybrid classifier [combining both rule-based knowledge and machine learning (ML) with the ML classification threshold set to 0.3] to separate the user’s correction utterance into distinct functional parts (see Figure 3): synonymous keywords and phrases for the various intents, deictic references, granularity of the modification, specific location (relative or absolute) versus range of modification, and conversational-styled elements. For training the classifier, we used 1,005 manually annotated training examples across all the 10 intents currently supported by our system (see Table 1). Once, *Dialogflow* labels the different functional categories, a grammar checker validates the well-formedness of the utterance based on their functional role within the utterance. If valid, the correction utterance triggers a relevant correction operation on the text.

[3] https://dialogflow.com/.

The command-based technique requires users to specify the intent and any accompanying parameter(s) as necessary. As an example of how the command-based technique works with the *Insert* intent, let us consider a sample insertion command: "**INSERT** *fox* **after** *brown.*" Here, "**INSERT**" is a keyword for the *Insert* intent, "*fox*" is a parameter to specify the content to be inserted into the text, and the phrase "**after** *brown*" is a parameter that specifies the location context. In this example, a relative location context is specified by the *spatial deixis*, "**after**," and a *target* word, "*brown.*" However, the user may chose to specify the location context in a more complex manner, e.g., the user says: "at the **end** of the sentence having the word *brown* in it, **INSERT** *lazy dog.*" *Dialogflow*'s hybrid classifier extracts the underlined section of the command as the location parameter and labels the various parts: "**end** of the sentence" as the *absolute target location*, "having the word" as the *container phrase*, "*brown*" as the *target* word, "**INSERT**" as the *keyword*, and "*lazy dog*" as the *content parameter* to be inserted. Once labeled and the grammar checker finds the command valid, the cursor is placed at the end of the specified sentence for the content parameter, "*lazy dog*," to get inserted as the last two words of the modified sentence.

To give the user a fine-grained control for specifying the location context (for all applicable intents), our system allows the user to (1) specify the granularity of the text to be edited: word or sentence; (2) match the target text inside a specific sentence, e.g., "in the sentence having the words *brown fox* in it..."; (3) use spatial deixes: **start**, **end**, **before**, and **after** to specify locations within (**start**/**end**) or outside (**before**/**after**) of a sentence; (4) use contextual references: previous, this, and next to locate a sentence in a given context, e.g., "**DELETE** the next sentence"; and (5) use ordinal numbers to locate a sentence, e.g., "**REPEAT** the first sentence." The user can combine multiple location specification constructs to form a complex location query, e.g., "at the **beginning** of the sentence having the words *jumps over* in it, **INSERT** the words *A quick.*" In that, our tool supports a much broader vocabulary as compared to the current vocabulary of eyes-free voice commands [11].

4.3 Re-Dictation Technique

In Study 1, we observed that participants often correct sentences by re-speaking over unwanted parts of the text with the correct wordings. For example, to correct "the quick *drown* fox jumps...," a participant says "quick *brown* fox," where "*brown*" should replace "*drown.*" The underlined words on either side of the correction have an exact match in the text and are called the *context* words with respect to the correction. To make a correction, participants may say context words either to the left of the correction, or to the right, or on both sides (as in the previous example). Although this approach of text correction has been explored in the visual context [34], we focus on an eyes-free adaptation of this technique, which we call *Re-dictation.*

The key challenge in implementing Re-dictation is aligning the correction with the part of the text that needs to be corrected (*target text*). Here, it should be noted that the aim of our implementation is not to find a precise re-dictation algorithm that beats the state-of-the-art but to adapt the re-dictation-based correction technique to the context of eyes-free. In that, our aim is to achieve a fairly usable alignment accuracy along with real-time error correction (minimal latency between the user's correction instruction and its execution) and support for interacting with the text eyes-free.

We combine two strategies for the text alignment: (1) limit the search space in which to locate the target text, and (2) use language analysis to determine the target text by computing similarity between the user utterance and the sentences within the search space (*candidate sentences*).

4.3.1 Limit the Search Space. An important heuristic to locate the target text in the eyes-free scenario is *timing*: From our pilot studies, we noticed that nearly 98% of the time, a correction was

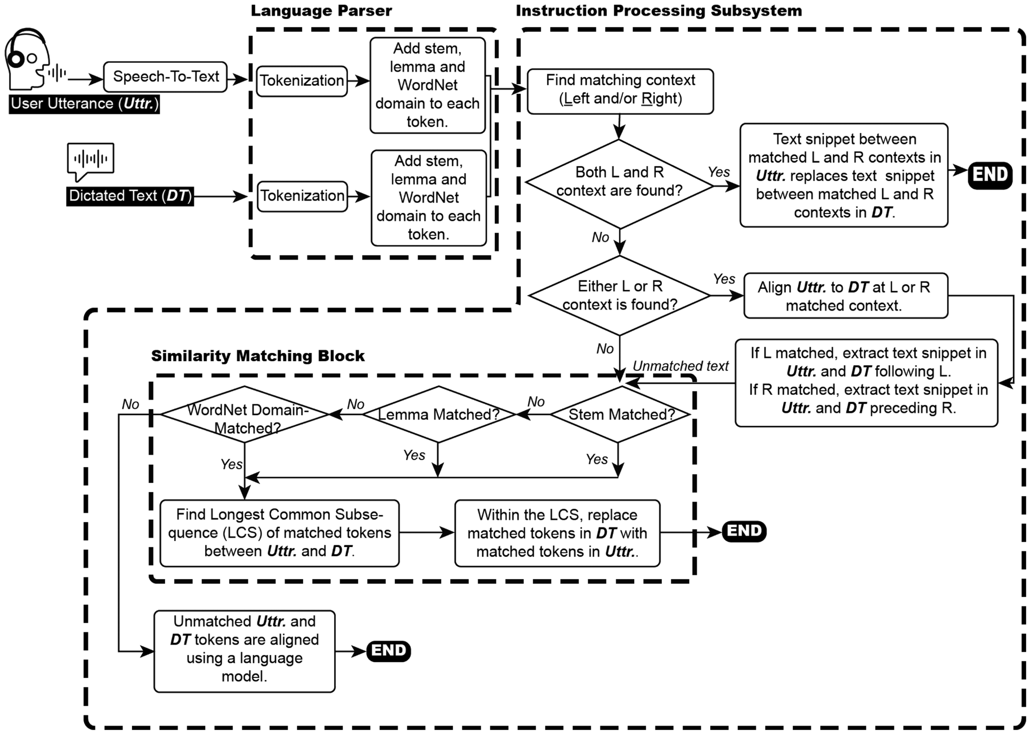
Language Parser
Instruction Processing Subsystem
Speech-To-Text
User Utterance (Uttr.)
Tokenization
Add stem, lemma and WordNet domain to each token.
Dictated Text (DT)
Tokenization
Add stem, lemma and WordNet domain to each token.
Find matching context (Left and/or Right)
Both L and R context are found?
Yes
Text snippet between matched L and R contexts in Uttr. replaces text snippet between matched L and R contexts in DT.
END
No
Either L or R context is found?
Yes
Align Uttr. to DT at L or R matched context.
If L matched, extract text snippet in Uttr. and DT following L. If R matched, extract text snippet in Uttr. and DT preceding R.
No
Unmatched text
Similarity Matching Block
Stem Matched?
No
Lemma Matched?
No
WordNet Domain-Matched?
No
Yes
Yes
Yes
Find Longest Common Subsequence (LCS) of matched tokens between Uttr. and DT.
Within the LCS, replace matched tokens in DT with matched tokens in Uttr..
END
Unmatched Uttr. and DT tokens are aligned using a language model.
END

Fig. 4. Flowchart for the Re-dictation technique design.

spoken within an average of 1.4 seconds of the target text being read out, a phenomenon referred to as "barge-in" in the literature [11]. The reason behind this barge-in phenomenon is straightforward: Due to the limited capacity of our short-term memory, if we do not immediately speak out the correction, we tend to forget the exact phrasing of the erroneous text. The timing heuristic allows us to limit the search space of the target text to the sentence that the user interrupts to speak a correction. Additionally, if a sentence is interrupted within the first four words (experimentally determined), the sentence preceding the interrupted sentence is also added to the search space; this is to accommodate the average time taken by a user to react to a correction cue towards the end of the preceding sentence. By limiting the search space, finding the target text within the body of the transcribed text becomes much easier and faster.

4.3.2 Determine the Target Text Using Language Analysis. Once the search space is truncated, the next step is to align the correction utterance to a part of the text within the search space. Figure 4 shows a high-level diagram of the system design. If the user provides context words both to the left and to the right of the intended correction, the alignment is trivial: The respective left and right contexts in the text and the utterance are aligned and the part between the contexts in the text is replaced by the part between the contexts in the utterance. As an example, to correct "The *slick round box* jumps over the lazy dog" if the user says, "The *quick brown fox* jumps," then the text and the utterance aligns at "The" (left context) and "jumps" (right context), and the portion of the *text* between the two contexts, i.e., "slick round box" is replaced by the portion of the *utterance* between the two contexts, i.e., "quick brown fox."

However, in other cases, the user may provide only the left or the right context, or may not provide any context at all. This makes determination of the part of the text that needs to be replaced (target text) nontrivial. In such cases, first, the text and the utterance are aligned at the matching context (if any), then using an in-house designed similarity matching algorithm, similar portions are identified and subsequently aligned. We use similarity as an alignment heuristic as the user-dictated correction is based on the initial user-dictated text and, hence, often share similar elements between them [8, 41]. For example, during the initial dictation (composition) phase, the user says "fox *leaped* over" and later wants to correct the verb "leaped." Here, the assumption is that the user would either correct to a different form of the verb with the same root such as "*leaps*" or "*had leaped*" (grammatically similar) or a different verb conveying an action with similar meaning such as "*jumped*" (semantically similar), or, in the case of speech-recognition error in the initial dictation, to something that is phonetically similar like "*tripped*" or "*skipped*." In our implementation, we check only for the first two similarities, i.e., grammatical and semantic similarities.

The **similarity matching algorithm** works in three hierarchical passes starting with *stem* matching followed by *lemma* matching and, finally, *semantic* matching. From stem matching to semantic matching, the criteria for computing a match are increasingly relaxed. Once a match is found in a pass, the matching algorithm terminates; otherwise, it continues with the remaining passes. Similarity matching starts with the language parser breaking both the candidate sentences and the user utterance into a series of tokens[4] using the Google Cloud Natural Language API.[5] This process is called *tokenization*. Once the tokens are extracted, the candidate sentence tokens are matched against the utterance tokens in the three passes mentioned before.

In the first pass, tokens match if they share the same stem. For example, "manage," "management," "managing," and "manager" share the same stem, "manag." If there are no matched tokens in the first pass, in the second pass, lemmas are matched. It is worthwhile to note that the lemma extraction or *lemmatization* is computationally more intensive than stem extraction or *stemming*. Stemming usually involves cutting off commonly occurring prefixes, suffixes, or derivational affixes from an inflected word without taking into account the context of the word usage. Therefore, stemming can neither differentiate between words that have different meanings depending on the part of speech nor can it identify words with different stems but conveying the same meaning. For example, although "go" and "went" convey the same idea in different tenses, stemming fails to capture this as the two words do not share a common stem. In contrast, lemmatization involves a thorough morphological analysis to understand the context of use of a word and, hence, can find the morphological root of the word. Referring back to the previous example, both "go" and "went" share the same lemma, "go." Other examples of words with different stems but having the same lemma are "is," "was," and "am" (lemma: "be"), "has" and "have" (lemma: "have"), and so on. Hence, lemma matching is more relaxed and can match words that were unmatched after stem matching if they share the same canonical/morphological root.

If the first two passes, i.e., stem and lemma matching, fail to find a match between the utterance and the text tokens, the third pass gets executed. In the third pass, the matching criterion is relaxed to finding semantic similarity between the two sets of tokens. Two tokens belong to the same semantic category if they share a common WordNet domain [20]. For example, "cat" and "dog" belong to the same WordNet domain: "noun.animal." Once a pass results in one or more pairs of matches, the matching stops and the longest common subsequence (LCS) of matched pairs is computed. The LCS determines both the target text and the part of the utterance that will replace the target text.

However, if utterance tokens remain unmatched even after all three passes of the similarity-matching algorithm, the unmatched utterance tokens are aligned to the text using a language

[4] A token is a string of contiguous characters between two spaces, or between a space and a punctuation.

[5] https://cloud.google.com/natural-language.

model derived from the Google Books Ngram Dataset (version 2).[6] This alignment is done using a simple linear interpolation [38] of bigrams and trigrams formed at all potential positions for the alignment and choosing the most probable position as the position of alignment (best alignment). However, to minimize potential recognition errors in the correction utterance from getting aligned, the alignment is rejected if the probability of best alignment is below an experimentally determined (by trial and error) cut-off = 0.1 (not shown in Figure 4 for simplification). For a rejected alignment, no error message is issued since a part of the utterance might already have been aligned in a previous step (if the utterance had either the left or the right context present, see Figure 4). Instead, the TTS resumes reading the text from either (1) just before the modified portion, as discussed before, if the correction utterance results in a modification of the text; or (2) just before the portion that was being read prior to the user interrupt, if the correction utterance does not result in a modification. In both cases, the user can unambiguously determine whether their intended modification has been performed.

5 STUDY 2: CONTROLLED EXPERIMENT—RE-DICTATION VERSUS COMMANDING

Our goal was to gain a deeper understanding of the pros and cons of the Re-dictation and Commanding techniques in editing text. To achieve this, we conducted a controlled experiment that compared these two techniques under a number of editing conditions. For the experiment, participants were asked to perform text-editing tasks eyes-free with each technique, following triggers of modifications embedded in the text. We designed tasks with varied complexities based on the number of edits per sentence and the relative positions between multiple edits. The experiment was to test our hypotheses and help us understand what parameters in text editing affect the use of each technique. Our hypotheses were as follows:

- H1: Re-dictation is *more efficient* to use than Commanding for making more complex edits.
- H2: Re-dictation is *easier* to use than Commanding for making more complex edits.
- H3: Commanding is *more efficient* than Re-dictation for making simpler edits.

5.1 Participants

A total of 16 paid participants (6 female, 10 male) were recruited for the experiment. The average age of the participants was 24 years (SD = 4.38). Ten participants were native English speakers. All the 16 participants were recruited from a local university and none of them had participated in Study 1.

5.2 Apparatus

The experiment was conducted in a quiet office. The participants used in-house-designed experiment software implemented with Node.js and JavaScript. The software was run on a locally hosted server on a MacBook Pro (2017 edition) with an Intel Core i5 dual-core processor running at 3.1 GHz using 16 GB of 2133 MHz LPDDR3 RAM and running macOS High Sierra (v10.13.6). Participants were provided with a Blue Yeti condenser microphone for voice input and a pair of Bose QC35 headphones for listening to the system audio. The experimenter had access to the same audio as the participant through a pair of Sennheiser CX180 earphones. The participant and the experimenter were seated face-to-face at the opposite long ends of a standard office desk, with the experimenter facing the laptop screen and the participant facing the microphone (Figure 5). The experimenter monitored the program running on the laptop, intervening only to use the next button to move through the different trials and start/stop a timer to measure the task-completion time.

[6] https://phrasefinder.io/documentation.

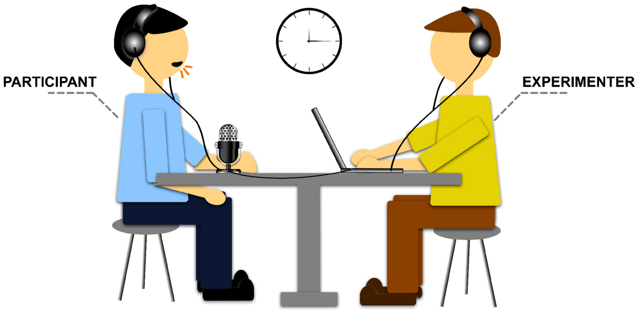

Fig. 5. Experimental setup for the controlled study. Participant had no visual engagement with the text.

There was no verbal communication between the participant and the experimenter throughout the duration of the experiment, except for the instructions and a final interview component. However, unlike Study 1, which was Wizard-of-Oz, restricting the participants from being able to see the experimenter was not necessary as the participants used a fully automated system. Moreover, a training session was conducted with the participants, prior to performing the actual experimental tasks, to get them familiar with the system. Also, the face-to-face arrangement allowed the participants to communicate with the experimenter using hand gestures to aid in the experimental procedure (explained further in Section 5.3).

5.3 Experiment Task

A trial of the task was to revise a test sentence (to-be-edited sentence) to match a given correct version of the sentence. For each trial, participants began by listening to two sentences: first, the correct sentence, followed by a test sentence that differed from the correct one in either one or two text positions. A one-word difference between the test sentence and correct sentence served as a *trigger* for correction. The participants were asked to edit the test sentence using a given technique. Participants could make multiple attempts to complete a trial. Also, they could opt to rehear the correct sentence at any time—as many times as needed—by requesting the experimenter with a hand gesture to replay the sentence. A trial ended in either of the following conditions: (1) a participant signaled the experimenter to move to the next trial; and (2) a participant made three consecutive failed attempts to correct the same error. The latter condition was to ensure that the experiment would not get stuck at a particular trial because of repeated errors in the recognition of the same word(s).

We operationalized the **Complexity** of edits with two factors. First, we defined the number of triggers (NumTriggers) embedded in one test sentence—*OneTrigger* and *TwoTriggers*. As one trigger required a participant to make one editing operation to correct it, two levels of NumTriggers gave us two types of tasks, each with a different level of complexity. For each of the two **Complexity** levels, different factors had to be defined to formalize the construction of experimental tasks.

5.3.1 *OneTrigger Tasks.*

A *OneTrigger* task was designed to use a test sentence with one out of three TriggerTypes (*Add*, *Delete*, *Replace*) embedded. The three TriggerTypes were evenly distributed across all the *OneTrigger* tasks and covered all our tested editing operations.

5.3.2 *TwoTriggers Tasks.*

From initial pilot studies, we found that with two triggers embedded in one sentence, both the position of the triggers and the types of edit operations required for the correction affect the editing complexity. Thus, based on our experience, we operationalized the

Table 2. Examples of Four TriggersPlacement with Two One-Word Correction Triggers (in Bold) in Each Test Sentence

TriggersPlacement	Test Sentence
Adjacent-Same Semantic (*AdjSame*)	Last night, | my **crazy brother** and I | had dinner | at a restaurant.
Adjacent-Diff. Semantic (*AdjDiff*)	Last night, | my friends and **me** | **having** dinner | at a restaurant.
NonAdjacent-Same Semantic (*NonadjSame*)	Last night, | my **Mom** and **Dad** | had dinner | at a restaurant.
NonAdjacent-Diff. Semantic (*NonadjDiff*)	Last night, | my **brother** and I | had **lunch** | at a restaurant.

The test sentences are segmented by *semantics*. For each test sentence, the correct sentence is: "Last night, my friends and I had dinner at a restaurant."

Complexity of a *TwoTriggers* task with two factors: TriggersPlacement and OperationCombination. For TriggersPlacement, we defined a *semantic segment* as the smallest unit in a sentence that can independently answer one of the *who*, *what*, *where*, *why*, *when*, and *how* questions. For instance, the sentence "<Joe><drove><to the railway station><at the outskirts of the city>" has four semantic segments. TriggersPlacement was defined based on whether the two triggers were embedded adjacent to each other (*Adj* and *Nonadj*) and whether or not they were within the same semantic segment (*Same* and *Diff*). Combining these two parameters gave us four types of TriggersPlacement: *NonadjSame*, *NonadjDiff*, *AdjSame*, and *AdjDiff* (as shown in Table 2). Furthermore, the types of edit operations required to correct each of the two triggers affect the editing complexity; for instance, it might be easier to combine two edits in one correction utterance if they require the same edit operation to correct it. Thus, we defined OperationCombination to distinguish these two situations: *SameOper* (*Add–Add*, *Delete–Delete*, *Replace–Replace*) and *DiffOper* (*Add–Delete*, *Add–Replace*, *Replace–Delete*). Together, they covered the full combination of the three editing operations.

To further ensure equal complexity of tasks under the same conditions, we constructed sentences for *OneTrigger* with four semantic segments with a mean length of 55.83 characters (SD = 3.65), and *TwoTriggers* sentences with four to five semantic segments with a mean length of 67.19 characters (SD = 3.41). For *OneTrigger*, the single trigger was always placed in the second or third semantic segment of the test sentence, while for *TwoTriggers*, the two triggers were placed either in the second and third, or in the third and fourth segments such that they satisfy the conditions of TriggersPlacement.

5.4 Design and Procedure

We employed a within-subject design for this experiment. As explained above, *Complexity* was operationalized by different factors for *OneTrigger* and *TwoTriggers* sessions. Therefore, we had *OneTrigger* sessions employing a 2×3 design with 2 levels of Technique (Commanding, Re-dictation) and 3 levels of TriggerTypes (*Add*, *Delete*, *Replace*); *TwoTriggers* sessions employed a $2 \times 4 \times 2$ design with three factors: Technique (Commanding, Re-dictation), TriggersPlacement (*NonadjSame*, *NonadjDiff*, *AdjSame*, *AdjDiff*), and OperationCombination (*SameOper*, *DiffOper*).

To better counterbalance the order effect between the two Technique conditions, participants first performed all the tasks for the *OneTrigger* and *TwoTriggers* sessions for a given technique and then moved to the second technique. Half of the participants performed Commanding first and Re-dictation later, while the other half performed Re-dictation first and Commanding later. For each Technique, participants were first briefed on the technique, after which they performed a training task with at least three trials (one each for add, delete, and replace operations) with one trigger embedded in the test sentence. Participants started the measured trials once they felt comfortable with the technique.

The measured trials for each Technique began with an *OneTrigger* session and continued with a *TwoTriggers* session for all participants. After finishing all the tasks for one Technique, participants filled out a NASA-TLX test [14] to assess the task load and a System Usability Scale (SUS) questionnaire [4] to assess the overall usability of that technique. At the end of the experiment, we interviewed the participants on their preferred technique and their subjective experiences.

For *OneTrigger* sessions, the presentation orders of TriggerTypes were counterbalanced across techniques and participants. For *TwoTriggers* sessions, the presentation orders for TriggersPlacement and OperationCombination were counterbalanced across participants using a Latin Square. In the end, there was one trial per combined condition, which gave us a total of 16 participants × 2 Technique × (*OneTrigger* [3 TriggerTypes] + *TwoTriggers* [4 TriggersPlacement × 2 OperationCombination]) = 384 trials. The entire experiment lasted for about an hour.

5.5 Data Collection

We collected the following. (1) *Task-Completion Time* (*TCT*): A timer was started when a participant started listening to the test sentence and was stopped on trial completion. (2) *NumAttempts*: number of attempts made by a participant to finish a trial. (3) *NumReplay*: number of times a participant requested the experimenter to replay the correct sentence. (4) *ErrorIndex*: Error index was calculated as the ratio of the edit distance between the correct and the *test* sentences to the edit distance between the correct and the *revised* sentences. Hence, $ErrorIndex = 0$ means that the revised sentence matches the correct sentence exactly, while $ErrorIndex \geqslant 1$ signifies that the revised sentence has the same ($ErrorIndex = 1$) or more number of errors ($ErrorIndex > 1$) than the unrevised test sentence. We used the Levenshtein distance [17], a commonly-used string metric, to compute the edit distance between the two strings. (5) *Subjective experience*: We collected the participants' perceived SUS ratings and weighted NASA Task Load Index ratings for each technique. Their subjective feedback was collected through semistructured interviews.

5.6 Results

5.6.1 Task-Completion Time.

H3 partially confirmed: Commanding is faster for single deletion, although not significantly faster. For *OneTrigger* conditions, a repeated-measures ANOVA was performed on *TCT* ~ Technique × TriggerTypes. No significant main effect was observed for Technique ($F_{1,15} = 2.03$, $p = .175$)—although *Deletion* with Commanding (20.31 ± 4.03) was faster than with Re-dictation (27.8 ± 11.17), the result was not found to be statistically significant. However, we found a significant main effect of TriggerTypes ($F_{2,30} = 3.95$, $p = .03$) on *TCT*, with effect size ($\eta^2 = 0.21$). Posthoc tests with Bonferroni correction revealed that *Delete* was significantly faster than *Add* ($p_{\text{bonf}} = .027$). Figure 6 shows the task-completion times with Commanding and Re-dictation for *OneTrigger* tasks.

H1 confirmed: Re-dictation is faster than Commanding for complex editing operations. For *TwoTriggers* conditions, a repeated-measures ANOVA was performed on *TCT* ~ Technique × TriggersPlacement × OperationCombination. We found significant main effects of Technique ($F_{1,15} = 36.56$, $p < .001$, $\eta^2 = 0.709$) and TriggersPlacement ($F_{3,45} = 4.23$, $p = .01$, $\eta^2 = 0.22$) on *TCT*. Whether the two embedded triggers were the same (*SameOper*) or different (*DiffOper*) had no significant effect on *TCT*. Furthermore, we found a significant Technique × TriggersPlacement interaction effect ($F_{3,45} = 6.23$, $p = .001$, $\eta^2 = 0.293$). To analyze this interaction, a pairwise analysis was performed between Technique (Commanding, Re-dictation) for each TriggersPlacement (*NonadjSame*, *NonadjDiff*, *AdjSame*, *AdjDiff*). Wilcoxon signed-ranks tests indicated that for *NonadjSame* ($W = 136, p < .001, r_{rb} = 1.0$) and *AdjDiff* ($W = 133, p < .001, r_{rb} = 0.956$), Re-dictation was significantly faster than Commanding (see Figure 7). The above effects

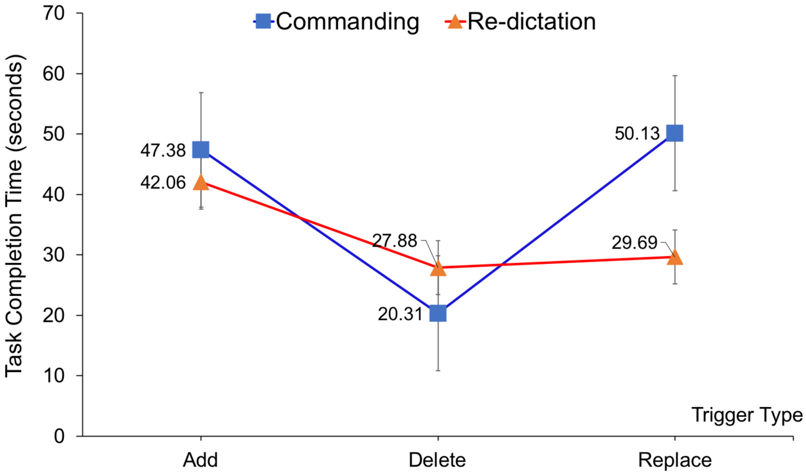

Fig. 6. *TCT* for *OneTrigger*—Technique × TriggerTypes.

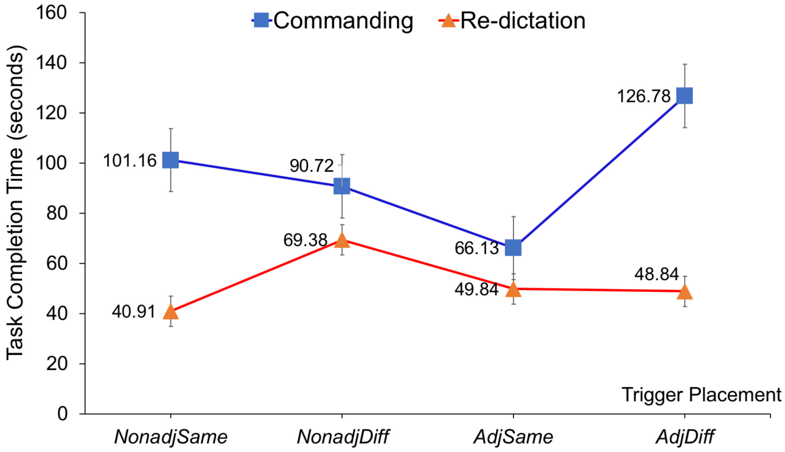

Fig. 7. *TCT* for *TwoTriggers*—Technique × TriggersPlacement with a significant interaction effect.

tell us that Re-dictation is faster than Commanding for editing multiple words that are in close *proximity* but in *different* semantic segments. If multiple words are in the *same* semantic segment, Re-dictation is faster than Commanding when the word positions are away from each other (not adjacent).

Both the placement of edits with respect to semantic segments and the distance between the edits matter.

On one hand, Re-dictation was significantly faster than Commanding for (1) *AdjDiff*, but not for *AdjSame* (Figure 7); and (2) *NonadjSame*, but not for *NonadjDiff*. This shows that even if the relative distance between two edited words is the same, whether or not these words are within the same semantic segment significantly affects the task-completion time. On the other hand, Re-dictation was significantly faster than Commanding for (1) *NonadjSame*, but not for *AdjSame*; and (2) *AdjDiff*, but not for *NonadjDiff* (Figure 7). This shows that even if the edited words are similarly positioned with respect to the semantic segments in the text, the distance between the edited words significantly affects the task-completion time.

To understand why, let us analyze each technique in more detail. To form an editing instruction with Commanding, users need to: (1) decide on the type of operation (add/delete/replace) that needs to be performed; (2) recall the syntax for making the intended operation; and (3) remember the target (to-be-modified) words in the sentence. Meanwhile, Re-dictation does not require users to remember or specify operations and target words. Users simply repeat part of the original sentence with the modifications included. Hence, if a correction spans multiple semantic segments, it is easier to make a Re-dictation utterance, which has the same grammatical structure (that of the English language) as the original text. However, a Commanding utterance needs to follow a particular command syntax. Consequently, the cognitive load of recalling information and framing a Commanding correction utterance gets compounded with an increase in the number of semantic segments that need to be corrected. While for simple modifications, it is still easy for users to perform Commanding and the difference between the two techniques is not obvious, but for more complex editing, the benefits of Re-dictation are enlarged and are more apparent.

In fact, we found that when the editing situation was more complex (such as *NonadjSame* or *AdjDiff*), it was difficult for participants to issue a single command to fix all the problems. As a result, participants mostly formed separate edit commands to edit multiple words, thereby leading to an increase in the task-completion time. Yet, with Re-dictation, it was still possible to complete all the corrections in one go. This finding is consistent with our analysis of the number of attempts that participants made to complete the editing tasks (see later).

5.6.2 Number of Attempts. A repeated-measures ANOVA was performed on *NumAttempts* ~ Technique × TriggerTypes for *OneTrigger* conditions and *NumAttempts* ~ Technique × TriggersPlacement × OperationCombination for *TwoTriggers* conditions. For *OneTrigger*, there was a significant main effect for Technique ($F_{1,15} = 7.15$, $p = .017$, $\eta^2 = 0.323$), with Re-dictation (1.63 ± 1.2) requiring a significantly fewer number of attempts to complete a task than with Commanding (2.15 ± 2.04). There was also a significant main effect of TriggerTypes ($F_{2,30} = 4.8$, $p = .016$, $\eta^2 = 0.243$) on *NumAttempts*, with *Delete* (1.125 ± 0.34) requiring a significantly fewer number of attempts than *Add* (2.41 ± 2.15) and *Replace* (2.125 ± 1.74). There was no significant Technique × TriggerTypes interaction effect.

For *TwoTriggers*, there was a significant main effect for Technique ($F_{1,15} = 27.74$, $p < .001$, $\eta^2 = 0.649$) with Re-dictation (2.32 ± 2.1) requiring a significantly fewer number of attempts than with Commanding (4.73 ± 3.39), and TriggersPlacement ($F_{3,45} = 3.21$, $p = .032$, $\eta^2 = 0.176$). Furthermore, a significant Technique × TriggersPlacement interaction effect was found on *NumAttempts* ($F_{3,45} = 7.62$, $p < .001$, $\eta^2 = 0.337$), similar to the one observed for *TCT* (Figure 7). The number of attempts is probably a main contributor for the effects found on *TCT*.

5.6.3 Number of Replays of the Correct Sentence.
H2 confirmed: Re-dictation is easier to use than Commanding for making longer and more complex edits.
A repeated-measures ANOVA was performed on the number of replays of the correct sentence that was requested by the participants during a task trial: *NumReplay* ~ Technique × TriggerTypes for *OneTrigger* conditions and *NumReplay* ~ Technique × TriggersPlacement × OperationCombination for *TwoTriggers* conditions. For *OneTrigger*, no significant main or interaction effects were observed. For *TwoTriggers*, there was a significant main effect for Technique ($F_{1,15} = 32.37$, $p < .001$, $\eta^2 = 0.683$) with participants requesting a significantly fewer number of replays with Re-dictation (0.45 ± 0.7) than with Commanding (1.23 ± 1.13). These findings suggest that Commanding imposes a higher demand for memorizing the text content.

We also observed a significant main effect for TriggersPlacement ($F_{3,45} = 4.71$, $p = .006$, $\eta^2 = 0.239$) on *NumReplay*. Posthoc tests with Bonferroni correction revealed that for *AdjSame* there was a significantly fewer number of replays than for *AdjDiff* ($p_{\text{bonf}} = .027$) and *NonadjDiff* ($p_{\text{bonf}} = .035$). There was also a significant main effect for OperationCombination ($F_{1,15} = 37.74, p < .001$, $\eta^2 = 0.716$) with *SameOper* (0.6 ± 0.93) having a significantly fewer number of replays than for *DiffOper* (1.08 ± 1.05). These findings suggest that if an edit operation involves editing words that span multiple semantic segments, then the need for memorizing the text content is higher than had the words been part of the same semantic segment.

5.6.4 Error Index. A repeated-measures analysis of variance (ANOVA) was performed on *ErrorIndex* ~ Technique × TriggerTypes for *OneTrigger* conditions and *ErrorIndex* ~ Technique × TriggersPlacement × OperationCombination for *TwoTriggers* conditions. There were no significant main or interaction effects, irrespective of the number of triggers. Though not significant, overall, Commanding had a fewer number of errors than Re-dictation for both *OneTrigger* (Commanding: .05 ± .16, Re-dictation: .21 ± .51) and *TwoTriggers* (Commanding: .25 ± .47, Re-dictation: .43 ± 1.69). These numbers suggest that Commanding allowed a more precise control over text modification than Re-dictation.

5.6.5 Subjective Experiences. A Wilcoxon signed-rank test for both SUS and NASA-TLX scores indicated that the perceived usability with Re-dictation (75.94 ± 14.3) was significantly higher ($W = 5.5, p = .006, r_{rb} = -0.919$) than with Commanding (55.47 ± 21.43). Also, the task load as measured using a weighted NASA-TLX was significantly lower ($W = 115, p = .016, r_{rb} = 0.691$) with Re-dictation (45.17 ± 15.8) than with Commanding (60.37 ± 24.05). These findings suggest that Re-dictation results in a lower task load and a higher perceived user experience. Thus, these findings further confirm H2 and is consistent with our findings from the quantitative measures.

Two major perceptions emerged from the participants' subjective feedback as follows:

(1) *Re-dictation feels more "natural"*: During the interviews, 11 of the 16 participants mentioned that Re-dictation felt more "natural" as it aligned to their mental model of how they "would correct a human-being." Moreover, seven participants mentioned that with Re-dictation, if the system failed to correctly align a correction utterance with the test sentence, they were rather forgiving and willing to include more words from the surrounding context (left/right context) in their subsequent correction attempt so as to guide the system better. On the other hand, four participants were not as comfortable repeating a command-based correction utterance. This is likely due to the fact that Re-dictation provides more alternatives for participants to correct an error; e.g., if the target is misaligned or a recognition error happens, the participants can help the system to correct it by providing more context words, and as we know, the system's recognition ability typically increases with more context, which often solves the problem. For Commanding, if the system makes a mistake, the only way to correct it is to repeat the same instruction, and if the system still fails to recognize it correctly, the participant has no other way to help the system, resulting in more frustration.

(2) *Commanding allows more control*: Although we discussed many benefits of Re-dictation, we cannot deny the value of Commanding. In fact, participants also recognized the advantages of the Commanding technique. A total of 10 out of the 16 participants suggested that command-based deletions align with their thought process while mentally conceiving a deletion: "*I need to delete this*" rather than "*this is how I want my correct sentence to be*" (P2). As P1 mentioned, "*Commands give you a (sic) precise control over what you want to change, with the second-approach (Re-dictation) you have to rely on the computer to decide how the change should look like (referring to the alignment).*" P12 mentioned, "*it (Commanding) is consistent even when it fails. I would, sort of, get the sense (of) when it would fail even before*

I got the error message (error feedback by the system). It's like you are in control and you are responsible for when it fails." Evidently Commanding, despite its inherent limitations, offers up a fine-grained control for editing text.

5.7 Summary

This experiment provided us an understanding of how users use Re-dictation and Commanding techniques for eyes-free text editing. While Re-dictation was more efficient and easier to use than Commanding for tasks that require more complex editing operations, Commanding performed similarly to Re-dictation for single operations and worked better for *deletions*. We identified two important factors in play: placement of the required edits with respect to the semantic segments of the text and the number of intervening words by which the required edits are separated. As mentioned earlier, we found a significant interaction effect of these two factors on the users' text-editing efficiency. Our findings from this experiment suggest that given both techniques have their unique advantages, supporting both techniques in a unified system might improve the usability and naturalness of user interaction with an eyes-free text-editing system. In particular, a unified system would allow the users a preferred choice of technique based on the edit length and complexity of the intended correction.

6 VOICEREV: COMBINING RE-DICTATION AND COMMANDING IN THE SAME SYSTEM

Since our goal was to assess the effects of each technique on text-editing interactions, the first iteration of our tool was designed to function with either of the two techniques individually. Yet, as suggested by the findings from Study 2, supporting both the techniques together in a unified system might improve the efficiency and user experience of editing text eyes-free (more under Section 8). To accommodate this, we developed in-house a voice-based tool—VoiceRev, a novel system to combine both Commanding and Re-dictation together in the same interface such that the user can seamlessly switch from one technique to the other without having to switch modes. With the techniques combined, our tool should classify an incoming user utterance as either a Commanding utterance or a Re-dictation utterance and execute the corresponding text-editing routine.

A schematic of the unified system is shown in Figure 8. First, the user utterance is cleaned off conversational-styled elements (if present) and checked for command keywords (or synonyms) similar to the design for the Commanding technique described earlier (Section 4.2). If a keyword is matched and associated parameters required to carry out a command-based editing operation could be found, the user utterance is classified as a Commanding utterance. Otherwise, the system treats the utterance as a Re-dictation utterance and performs similarity matching to compute potential alignment positions (Section 4.3.2). To reduce the number of false positives while classifying an utterance as a Re-dictation utterance, if the maximum alignment probability across all potential alignment positions is below a certain threshold = 0.3 (experimentally determined through trial and error), no editing action is performed and an appropriate voice error feedback is provided to the user.

7 STUDY 3: EVALUATING VOICEREV—RE-DICTATION WITH COMMANDING

Based on our findings from Study 2, we had hypothesized that combining both Commanding and Re-dictation in a single coherent interface would improve the usability of eyes-free text-editing systems. To test this hypothesis, we implemented a voice-based system that combines the two techniques. Our objective in this study was to evaluate the unified system with realistic tasks where participants would use speech to first compose a piece of text and then revise it using our

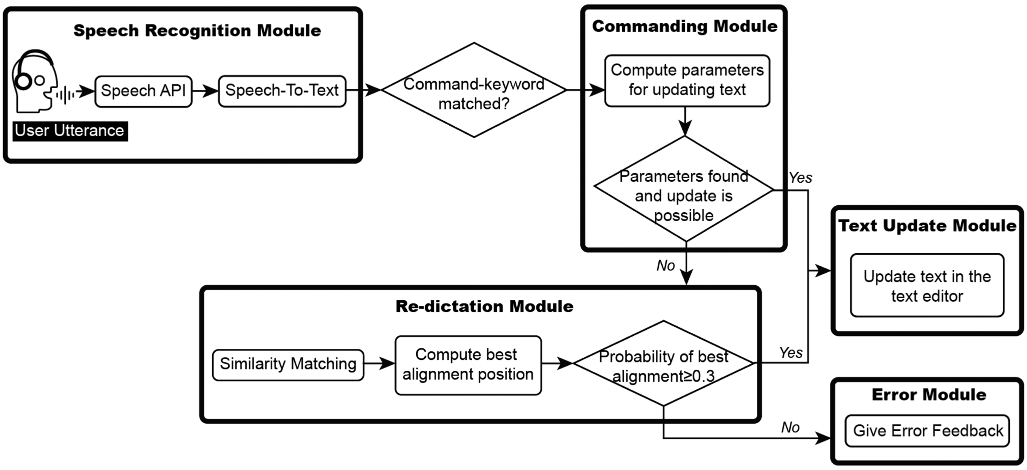
Speech Recognition Module
User Utterance
Speech API
Speech-To-Text
Command-keyword matched?
Commanding Module
Compute parameters for updating text
Parameters found and update is possible
Yes
No
Text Update Module
Update text in the text editor
Re-dictation Module
Similarity Matching
Compute best alignment position
Probability of best alignment≥0.3
Yes
No
Error Module
Give Error Feedback

Fig. 8. Design of the unified system combining both Commanding and Re-dictation—system components and information flow.

system. This setting also represents one of the promising application scenarios where we foresee our system being used.

7.1 Participants

Eight paid participants (four female, four male; age mean = 23.3 years, SD = 3.1) were recruited from within the university community. Three were native English speakers. Two participants were occasional users of voice-based assistants (one a Siri user and the other an occasional user of Amazon's Alexa assistant on an Echo Dot device), whereas none of them had any prior experience with voice-based dictation or correction systems. None of the participants had taken part in either of the two previous studies.

7.2 Apparatus

With the exception of the text-editing interface for which we used our unified system, VoiceRev, the rest of the apparatuses used were exactly the same as the ones used for Study 2.

VoiceRev was designed to operate in two distinct modes: *composition* and *revision*, with manual mode switching. At the start of the experiment, participants were asked to compose a piece of text using speech. For this, the system was set to the *composition* mode, which allowed composition of the text real-time using the system's voice-dictation feature. Also, the system allowed the experimenter to add punctuation to the text even during the live transcription process. Using this feature, the experimenter added the relevant punctuation by placing commas and sentence end marks (period, question mark, or exclamation mark) at appropriate locations in the text. In the composition mode, the system does not interpret any user utterance as a correction utterance, but simply transcribes the dictated text into a text editor. After composing the text, participants revised the text using the *revision* mode. The mode switching from composition to revision was done manually by the experimenter. Unlike the composition mode, in the revision mode, the system interprets every user utterance as a correction utterance.

Furthermore, in the revision mode, participants could choose between Commanding or Re-dictation to make a correction. The system was designed to understand a correction utterance as either Commanding or Re-dictation based on *how* the utterance was framed, without the need for explicitly switching from one technique to the other. Hence, in effect, to revise a composed block of text, a participant could use Commanding for some corrections and Re-dictation for the others.

7.3 Task

The task for the participants was to first compose a piece of text using voice dictation (six to eight sentences) and then revise the dictated content eyes-free. Participants were instructed to revise the previously dictated text to its best version possible, i.e., free from logical and grammatical mistakes, having a clear and coherent flow, and depending on the type of content, publishable either on social media or as a more formal document, e.g., a formal e-mail to a professor.

Before the composition phase began, participants were given a list of topics to choose from, either formal or informal in nature. The topics were selected to encourage the participants to speak freely: a movie/TV series review, an e-mail to a professor/industry seeking summer internship opportunities, an e-mail to the housing management with a complaint, a recent travel experience, and a recent current affair that had caught the participant's attention. If a participant did not feel comfortable with any of the given topics, they were free to decide on a topic of their choice. After selecting a topic, participants could optionally take 30 seconds to mentally prepare an outline of what they would talk about during the composition phase.

7.4 Design and Procedure

Each participant took two trials of the composition-and-revision task, each trial with a different topic. For each trial, a participant would start by choosing a topic and then perform the *composition* before finally moving on to the *revision*. For the revision, a participant could make multiple correction utterances and revise as many regions of text as was deemed necessary by the participant.

Before starting with the task trials, participants were given a training session to familiarize themselves with both the Commanding and the Re-dictation techniques of text editing. The training session was followed by a warm-up session where the participants composed a short text and then revised it using the two techniques. The warm-up session ended when the participants felt confident in using both the techniques.

After the two task trials, we conducted a short semistructured interview for 10 minutes. The entire experiment lasted for approximately 70 minutes. Both the interview (audio) and the task trials (screen and audio) were fully recorded with the participants' informed consent.

7.5 Data Collection

For every correction utterance made by a participant, we recorded the technique used (Commanding or Re-dictation). In the case of a Commanding correction utterance, we recorded the correction operation attempted (*Add*, *Delete*, or *Replace*), and the number of words that were modified by a correction utterance, *NumWords*.

We also measured two derived counts: the number of failed attempts while making a command-based correction—*NumCommandFails* and the number of failed attempts while making a re-dictation-based correction—*NumRedictFails*. In our system, we had implemented a feature to label a correction utterance as "failed" with a simple key-press. During the experiment, the experimenter would label a participant's utterance as failed if the system failed to perform what was intended by the participant either due to a *recognition error* or due to a *system error*. A recognition error would occur if the speech recognizer misrecognized a participant's utterance in part or in whole. For example, if the participant said "**DELETE** *buy*," but the system picked it up as "**DELETE** *by*," then the deletion command would fail if "by" was not present in the text. On the contrary, a system error would occur if the speech recognition was as intended by the participant, but the action performed by the system was different from the participant's

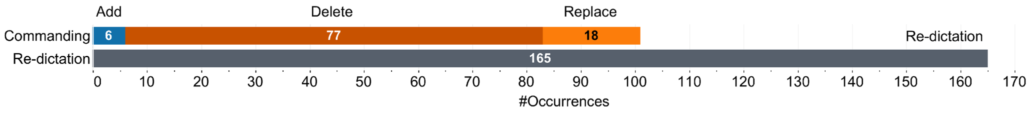

Fig. 9. Count distribution of successful correction utterances.

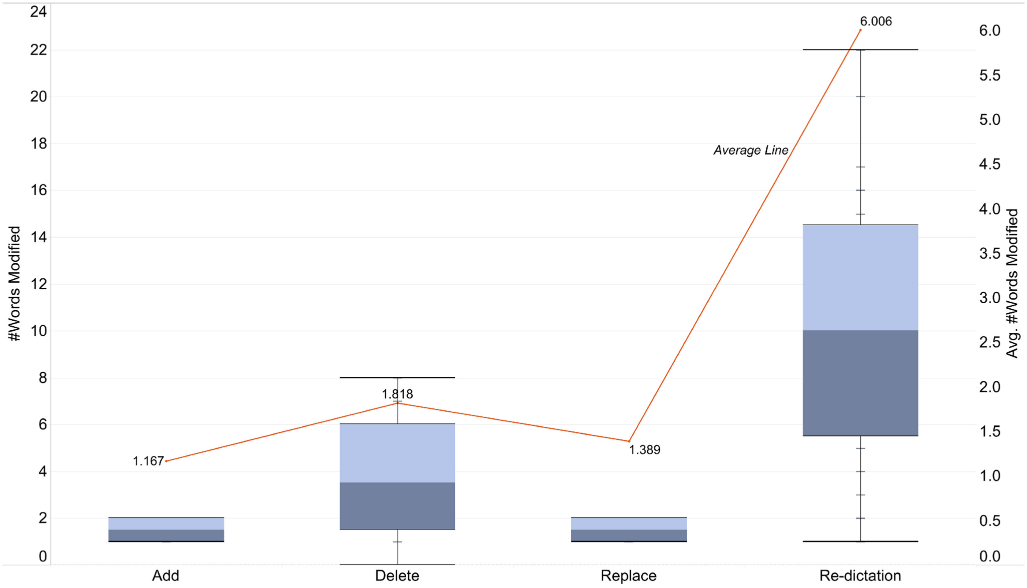

Fig. 10. Comparison of the number of words modified with the Commandingand Re-dictation operations. An average line is plotted to show the average number of words modified by each operation.

desired action. For example, to insert the missing word "*fox*" after "*brown*" in the sentence: "The quick brown jumped over the lazy dog", the participant re-dictated the erroneous portion as "fox jumped over"; however, the system misaligned the correction utterance with the text resulting in the corrected text, "The fox jumped over the lazy dog." This would result in a system error as the alignment determined by the system was different from what was intended by the participant.

7.6 Results

After discarding utterances where the participants did not attempt any insertion, deletion, or replacement (e.g., read/repeat or undo/redo requests), 64 *NumCommandFails* and 39 *NumRedictFails*, we collected a total of 266 successful correction utterances across 8 participants. These 266 utterances consisted of 165 Re-dictation utterances, 77 command-based deletions, 18 command-based replacements, and 6 command-based insertions (Figure 9).

Both the mean and the variability for *NumWords* with Re-dictation (6.01 ± 3.88) were much higher than any of the command-based operations (see Figure 10). Mean of *NumWords* for command-based *delete* (1.82 ± 1.74) was higher than both *replace* (1.39 ± 0.5) and *add* (1.17 ± 0.41). The higher mean for Re-dictation corroborates our findings from Study 2 that Re-dictation is more suited to and the preferred technique for making longer and more complex changes to the text. Furthermore, the higher variability for Re-dictation combined with the low number of command-based replacements and insertions (only about 9% of the total number of successful edit operations) shows the participants' preference in using Re-dictation for also making shorter

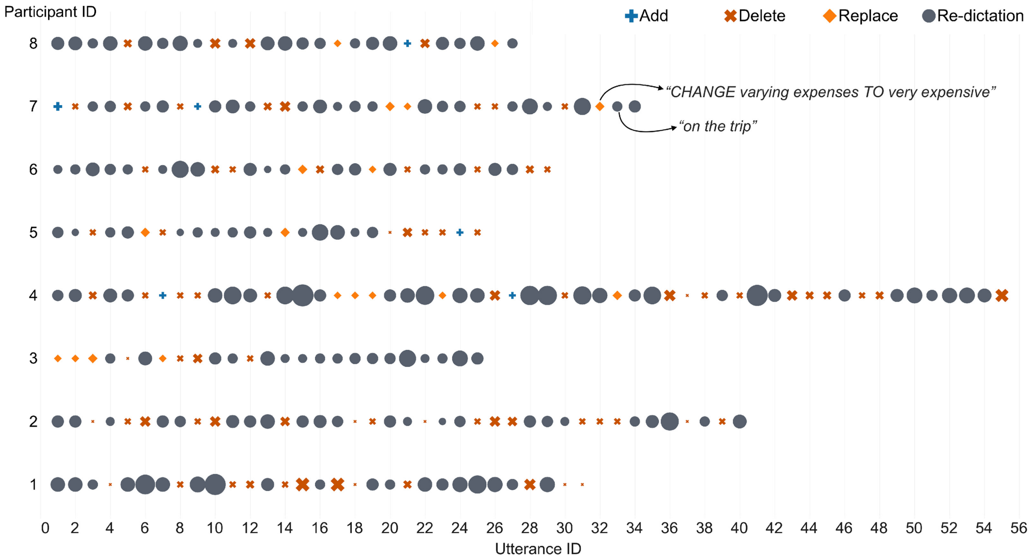

Fig. 11. Timeline of participants' successful correction utterances. Size of each symbol, representing an edit operation, is proportional to the number of words modified in that operation, i.e., *NumWords* (size is linearly scaled between 0 and 22). Labeled are two utterances from P7 showing consecutive Commanding and Re-dictation utterances.

edits, especially for insertions and replacements. Command-based deletions, however, made up ≈29% of the total number of successful edit operations, echoing our findings from Study 2 that Commanding is easier and the more suitable technique for deleting shorter segments of text.

Figure 11 shows a timeline of participants' correction utterances (failed attempts not shown). Two consecutive utterances from P7's data (utterance IDs 32 and 33) have been labeled on the figure as sample utterances—the first is a command-based change utterance and the second a Re-dictation utterance. This shows that participants could implicitly switch between techniques just by altering the framing of the utterances.

7.6.1 False Positives. Of the 165 Commanding utterances (101 successful, 64 failed), 22 utterances were misclassified as Re-dictation utterances. The reason for these misclassifications was mostly speech-recognition error in detecting the command keyword. These misclassifications were counted as false positives for Re-dictation (≈13.3%). The percentage of false positives for Commanding (Re-dictation utterances misclassified as Commanding) was much smaller (≈7.35%).

False-positive utterances were confusing for the participants as they would not be sure as to why a Commanding utterance behaved as a Re-dictation utterance or vice versa. However, the recovery was not difficult. For example, false positives for Commanding (i.e., correction utterance was originally intended to be a Re-dictation utterance) mostly did not conform to the Commanding grammar structure and lacked the well-formedness of a valid command. Consequently, the command would fail without changing the text. To recover, participants would simply repeat the Re-dictation correction utterance or include more context words in their repeated correction attempt. On the other hand, false positives for Re-dictation would cause unwanted alterations in the text, but with the ability to perform an *undo* operation, it was easy for participants to recover from the error.

In summary of this study, first, our approach of combining the two techniques in one system proved promising given the positive feedback about its usability. In the interview, P4 mentioned, "*It's great that I could use the two methods (techniques) interchangeably (without explicit switching). Actually, I was not thinking about what method I am (sic) using, I just said what came to me naturally.*" P2 mentioned, "*While deleting something, saying 'delete this' or 'delete that' felt more natural than repeating that part (correction by Re-dictation). For others (other operations), I felt re-dictating it felt way more (sic) easier.*" Secondly, the results from this study confirmed the ecological validity of our findings from Study 2. Re-dictation was overall the preferred technique for making longer and more complex (complexity as defined in Section 5.3 of Study 2) corrections, whereas for deleting shorter segments of text, command-based deletion was preferred to using Re-dictation.

8 DISCUSSION

In this section, we put our findings from Studies 2 and 3 in perspective and discuss the implications for the design of future voice-based text-editing interfaces.

8.1 Why Should Future Systems Support Re-Dictation?

Results from Studies 2 and 3 showed that Re-dictation is more efficient overall and easier to use, especially for making longer edits and more complex operations. In transcribing dictated speech, the automated speech transcription suffers from recognition errors even with state-of-the-art speech recognition. Oftentimes, the wordings of these errors are difficult to detect from the TTS audio of the transcribed text and, thus, difficult to speak out in a correction utterance. Unlike Commanding, Re-dictation does not involve speaking out the erroneous words for making a correction. This not only reduces the need for recalling the content, but is also very useful in fixing the recognition errors that would otherwise be very difficult to fix without a visual feedback of the text. In such situations, the interaction gets stuck or falls in an error-correction loop if the user can only use Commanding. Although, Re-dictation can be used with a screen as well, its benefits are even more apparent in eyes-free scenarios.

8.2 Users' Choice between Commanding and Re-Dictation

Although both Studies 2 and 3 revealed high-performance benefits of Re-dictation, we noticed that in the Wizard-of-Oz study, a smaller percentage of correction utterances were re-dictation utterances as compared to command-based utterances. This discrepancy might partly be attributed to the users' existing mental model of how a computing system should be instructed—using specific commands. Also, the inherent rigidity in how a command should be framed to be interpreted correctly gives users a sense of precise control in making the desired changes to the text. Hence, with no introduction to any particular technique, participants of the Wizard-of-Oz study demonstrated a preference for using commands. In contrast, Study 3 showed users' preference for using more re-dictation utterances. This might be due to a prior introduction to the Re-dictation technique via instructions and training. While the change in user behavior was likely due to an enhanced understanding of what is possible, further studies are needed to investigate how exactly the mental model of existing technology can shape users' unconscious behavior of adopting innovative technologies.

Furthermore, as the results of Studies 2 and 3 showed, command-based utterances are suitable for one-word edit operations, especially deletions. Combined with the users' existing mental model of using commands for giving instructions to a system, a voice-based text-editing interface that does not take into consideration such a mental model might frustrate some users, at least during the on-boarding stage. Therefore, we suggest the inclusion of both Commanding and Re-dictation in future systems to support voice-based editing of text.

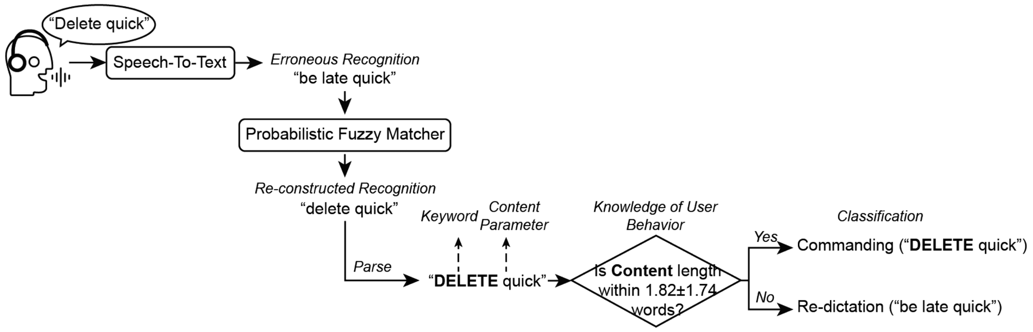
"Delete quick"
Speech-To-Text
Erroneous Recognition
"be late quick"
Probabilistic Fuzzy Matcher
Re-constructed Recognition
"delete quick"
Parse
Keyword
Content Parameter
"DELETE quick"
Knowledge of User Behavior
Is Content length within 1.82±1.74 words?
Yes
No
Classification
Commanding ("DELETE quick")
Re-dictation ("be late quick")

Fig. 12. Example of how the knowledge of user behavior can be leveraged to improve classification accuracy.

8.3 How to Combine Both Commanding and Re-Dictation in One Interface?

Unlike a system supporting only one technique, either Commanding or Re-dictation, a system supporting both the techniques first needs to classify a correction utterance as either Commanding or Re-dictation. The accuracy of the classification determines the success of the correction utterance in making the desired correction. The probability of an accurate classification decreases if the speech recognition is not accurate or if the grammar to interpret the correction utterance is relaxed, e.g., to support conversational-styled elements in the utterance. Our work provides insights into the user behavior, which can be used to improve the classification accuracy. Figure 12 shows an example of how classification accuracy can be improved by our enhanced knowledge of the user behavior.

In the example, the user says a correction utterance, "**Delete** *quick*." However, a recognition error results in the transformed utterance, "be late quick." With no command keyword present, the utterance might be misclassified as a Re-dictation utterance. Here, our knowledge of user behavior from Study 3 can be leveraged to get the correct classification. To elaborate, computing fuzzy string matching or phonetic similarity on the misrecognized utterance tokens against a list of valid keywords would reconstruct the utterance as "delete quick," which upon parsing separates the utterance into a keyword (**delete**) and a content parameter (*quick*). We know from Study 3 that users prefer command-based deletion for deleting text segments of length 1.82 ± 1.74 words. This knowledge can be leveraged to rightly classify the utterance as Commanding utterance despite the initial misrecognition. Thus, the text would be modified as was initially intended by the user.

9 LIMITATIONS AND FUTURE WORK

We have used state-of-the-art Google Cloud Speech Recognition API to conduct our study, but there are still recognition errors that affect the user experience of voice-based text editing. In our studies, we have tried to balance the number of native English-speaking participants to the number of nonnative English-speaking participants and believe that our results are fairly robust against recognition errors. However, in the future, if the recognition accuracy is improved, the results of our study should be verified. Also, currently, our system does not support disambiguation between homophones (words with the same pronunciation but different meanings, e.g., to, two, and too). Since this limitation affects both our studied techniques, we believe that our findings from the relative performance and usability comparison between the two techniques would apply to future systems independent of their ability to support homophone disambiguation. Nonetheless, we suggest that future explorations devise systems to identify and disambiguate between homophones and study if this enablement affects the two techniques differently.

Furthermore, our implementations of both the Commanding and the Re-dictation techniques are based on the English language. However, since both these techniques support word-level editing, their use can be extended to even logographic writing systems like Chinese and Japanese. In logographic writing systems, a logogram is a *written* character or glyph that represents an entire word or phrase. So, essentially logograms function as words that form the units of the *spoken* language, thus making our techniques relevant to such languages. Yet, as our findings from the comparative analysis of Commanding and Re-dictation are based on the English grammar and its underlying semantic structure, the findings must be verified before applying to languages with a different grammar structure.

In addition, our unified system combining the Commanding and Re-dictation techniques allows the user to modelessly switch between the two techniques. However, the implicit switching might sometimes result in misclassification of the technique based on how the correction utterance was framed or recognized. Future works should explore more sophisticated classification algorithms to improve the classification accuracy of users' correction utterances. Moreover, cases of misclassification can be confusing for the users. Future research should explore ways to better inform the users about the system's performed classification, say, by assigning distinct nonspeech audio sounds for Commanding and Re-dictation. Furthermore, currently, alignment computation for re-dictated utterances is done based on the 1-best recognition result of the speech recognizer. The alignment accuracy might be improved by a deeper integration of the alignment computation with the results from the speech recognizer, say, by considering the *N*-best list of results instead of just the 1-best result.

Furthermore, the composition and revision of text that our system supports are differentiated by manual switching between the two modes. This is in keeping with the scope of our current research, which was to explore only revision techniques. Yet, in a more realistic setting, users may constantly switch back and forth between composing and revising text. For example, a user may compose only a part of the text and want to revise it before moving on to compose the rest of the text. Future research can explore the user behavior of switching between the two modes and find more sophisticated mode-switching techniques.

Finally, future work should explore multimodality in input/output to further reduce the cognitive load of editing the text eyes-free. Multimodal input might include voice with gesture input, while multimodal output may explore audio with ambient and glanceable displays, among other techniques. Multimodality can offload some of the information that is currently being conveyed using only the audio and the voice channel, and may further improve the usability of text-editing interactions without the requisite of maintaining constant visual engagement with the text.

10 CONCLUSION AND IMPLICATIONS

With the advancement of speech recognition and machine learning technologies, voice-only interfaces are likely to play a more important role in HCI and in our lives. However, the utility of such interfaces would likely be limited if they cannot support real-time modification of the users' spoken content. This research explores how to better design a voice interface for users to edit text eyes-free. Informed by the naturally emerging user behavior from a Wizard-of-Oz study, we implemented and compared two eyes-free text-editing techniques: *Commanding* and *Re-dictation*, and identified their pros and cons in different task scenarios. As an optimal solution, we combined the two techniques in a novel unified system, VoiceRev, and evaluated it with realistic tasks. Results confirmed the ecological validity of our previous findings.

Hence, our studies provide convincing evidence that despite the challenges arising from the lack of visual feedback, we can improve the user experience of eyes-free text editing by combining Commanding and Re-dictation techniques in a single system. However, the implications of our findings extend beyond eyes-free scenarios. As suggested by Ghosh et al.'s *EYEditor* [12], a smartglass-based system that uses voice input to correct text on the go, both the user experience and the efficiency of text editing were greatly enhanced by allowing users to correct the text by re-speaking over erroneous parts with additional support for making precise command-based deletions in the text. Also, EYEditor's voice design allowed the users to better multitask than with a smartphone—with EYEditor, the users could maintain better awareness of their walking path while correcting the text. These findings suggest that our current research can be extended to a visual user interface to improve the user experience of on-the-go text editing.

Furthermore, our techniques and findings can be used in a variety of applications other than just text-processing software, e.g., in applications designed for Intelligent Personal Assistants (IPAs) and Voice User Interfaces (VUIs). A concrete use case of where our findings can be applied might be to make inline corrections to voice commands in VUIs—say, a Google Home user wants to set a calendar event and speaks out a title for the event, but the title gets misrecognized; the user on listening to the audio confirmatory feedback of the misrecognized title can quickly make a correction by Re-dictation or Commanding, based on the length and/or complexity of the correction.

ACKNOWLEDGMENTS

We thank Yang Chen for her generous help with designing some of the figures in the manuscript and Vanitha S. for designing Figure 5.

REFERENCES

[1] Dina Abdelrazik. 2017. *Enabling Voice in the Smart Home: A Parks Associates Whitepaper Developed for ULE Alliance.* Parks Associates Technical Report. Parks Associates, Addison, TX.

[2] Amazon.com. 2020. Amazon.com: Customer Reviews: Mail Box. Retrieved March 26, 2020 from https://www.amazon.com/product-reviews/B01LMLFNB6.

[3] Shiri Azenkot and Nicole B. Lee. 2013. Exploring the use of speech input by blind people on mobile devices. In *Proceedings of the 15th International ACM SIGACCESS Conference on Computers and Accessibility.* ACM, New York, NY, 1–8. DOI: https://doi.org/10.1145/2513383.2513440

[4] John Brooke. 1996. SUS-A quick and dirty usability scale. *Usability Evaluation in Industry* 189, 194 (1996), 4–7.

[5] 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 *Proceedings of the 2012 IEEE International Conference on Acoustics, Speech and Signal Processing.* IEEE, 4973–4976. DOI: https://doi.org/10.1109/ICASSP.2012.6289036

[6] Jason W. Clark, Rathinavelu Chengalvarayan, Timothy J. Grost, Dana B. Fecher, and Jeremy M. Spaulding. 2011. Voice dialing using a rejection reference. US Patent No. 8,055,502.

[7] Jan Cuřín, Martin Labský, Tomáš Macek, Jan Kleindienst, Hoi Young, Ann Thyme-Gobbel, Holger Quast, and Lars König. 2011. Dictating and editing short texts while driving: Distraction and task completion. In *Proceedings of the 3rd International Conference on Automotive User Interfaces and Interactive Vehicular Applications.* ACM, 13–20. DOI: https://doi.org/10.1145/2381416.2381418

[8] Catalina Danis and John Karat. 1995. Technology-driven design of speech recognition systems. In *Proceedings of the 1st Conference on Designing Interactive Systems: Processes, Practices, Methods, & Techniques.* ACM, 17–24. DOI: https://doi.org/10.1145/225434.225437

[9] Christopher Frauenberger and Tony Stockman. 2009. Auditory display design—an investigation of a design pattern approach. *International Journal of Human-Computer Studies* 67, 11 (2009), 907–922. DOI: https://doi.org/10.1016/j.ijhcs.2009.05.008

[10] Ira A. Gerson and Brett L. Lindsley. 1989. Method for entering digit sequences by voice command. US Patent No. 4,870,686.

[11] 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.* ACM, 1–10. DOI: https://doi.org/10.1145/3173574.3173977

[12] 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.* ACM. DOI: https://doi.org/10.1145/3313831.3376173

[13] 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 *Proceedings of the Human–Computer Interaction—INTERACT'99*. 133–140. DOI: https://pdfs.semanticscholar.org/0e6d/6a06f5b5783ea82597f48b45d6261bfa00cf.pdf

[14] Sandra G. Hart and Lowell E. Staveland. 1988. Development of NASA-TLX (task load index): Results of empirical and theoretical research. In *Advances in Psychology*. Vol. 52. Elsevier, 139–183. DOI: https://doi.org/10.1016/S0166-4115(08)62386-9

[15] Per Ola Kristensson and Keith Vertanen. 2011. Asynchronous multimodal text entry using speech and gesture keyboards. In *Proceedings of 12th Annual Conference of the International Speech Communication Association*. ISCA Speech. DOI: https://www.isca-speech.org/archive/interspeech_2011/i11_0581.html

[16] Nicole Yankelovich and Jennifer Lai. 1998. Designing speech user interfaces. In *Conference Summary on Human Factors in Computing Systems (CHI'98)*. ACM, 131–132. DOI: https://doi.org/10.1145/632716.632793

[17] Vladimir I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet Physics Doklady*, Vol. 10. American Institute of Physics, 707–710.

[18] Ewa Luger and Abigail Sellen. 2016. Like having a really bad PA: The gulf between user expectation and experience of conversational agents. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 5286–5297. DOI: https://doi.org/10.1145/2858036.2858288

[19] Arthur E. McNair and Alex Waibel. 1994. Improving recognizer acceptance through robust, natural speech repair. In *Proceedings of the 3rd International Conference on Spoken Language Processing*. ISCA Speech. DOI: https://www.isca-speech.org/archive/icslp_1994/i94_1299.html

[20] George A. Miller. 1995. WordNet: A lexical database for English. *Communications of the ACM* 38, 11 (1995), 39–41. DOI: https://doi.org/10.1145/219717.219748

[21] Don Norman. 2013. *The Design of Everyday Things: Revised and Expanded Edition*. Basic Books.

[22] Jun Ogata and Masataka Goto. 2005. Speech repair: Quick error correction just by using selection operation for speech input interfaces. In *Proceedings of the 9th European Conference on Speech Communication and Technology*. ISCA Speech. DOI: https://www.isca-speech.org/archive/interspeech_2005/i05_0133.html

[23] Sharon Oviatt. 2000. Taming recognition errors with a multimodal interface. *Communications of the ACM* 43, 9 (Sept. 2000), 45–51. DOI: https://doi.org/10.1145/348941.348979

[24] Sharon Oviatt and Robert VanGent. 1996. Error resolution during multimodal human-computer interaction. In *Proceedings of the 4th International Conference on Spoken Language*, Vol. 1. IEEE, 204–207. DOI: https://doi.org/10.1109/ICSLP.1996.607077

[25] Ian J. Pitt and Alistair D. N. Edwards. 1996. Improving the usability of speech-based interfaces for blind users. In *Proceedings of the 2nd Annual ACM Conference on Assistive Technologies*. ACM, 124–130. DOI: https://doi.org/10.1145/228347.228367

[26] Teresa L. Roberts and Thomas P. Moran. 1983. The evaluation of text editors: Methodology and empirical results.*Communications of the ACM* 26, 4 (1983), 265–283. DOI: https://doi.org/10.1145/2163.2164

[27] Johan Schalkwyk, Doug Beeferman, Françoise Beaufays, Bill Byrne, Ciprian Chelba, Mike Cohen, Maryam Kamvar, and Brian Strope. 2010. "Your word is my command": Google search by voice: A case study. In *Advances in Speech Recognition*. Springer, Boston, MA, 61–90. DOI: https://doi.org/10.1007/978-1-4419-5951-5_4

[28] 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*. ACM, 555–560. DOI: https://doi.org/10.1145/2388676.2388793

[29] Matthias Sperber, Graham Neubig, Christian Fügen, Satoshi Nakamura, and Alex Waibel. 2013. Efficient speech transcription through respeaking. In *Proceedings of the Interspeech*. ISCA Speech, 1087–1091. Retrieved from https://ahcweb01.naist.jp/papers/conference/2013/201308_INTERSPEECH_Sperber_01/201308_INTERSPEECH_Sperber_01.paper.pdf.

[30] Amanda Stent, Ann Syrdal, and Taniya Mishra. 2011. On the intelligibility of fast synthesized speech for individuals with early-onset blindness. In *Proceedings of the 13th International ACM SIGACCESS Conference on Computers and Accessibility*. ACM, 211–218. DOI: https://doi.org/10.1145/2049536.2049574

[31] Lisa J Stifelman, Barry Arons, Chris Schmandt, and Eric A Hulteen. 1993. VoiceNotes: A speech interface for a handheld voice notetaker. In *Proceedings of the INTERACT'93 and CHI'93 Conference on Human Factors in Computing Systems*. ACM, 179–186. DOI: https://doi.org/10.1145/169059.169150

[32] Bernhard Suhm, Brad Myers, and Alex Waibel. 2001. Multimodal error correction for speech user interfaces. *ACM Transactions on Computer–Human Interaction* 8, 1 (2001), 60–98. DOI: https://doi.org/10.1145/371127.371166

[33] Markku Turunen, Jaakko Hakulinen, Nitendra Rajput, and Amit A. Nanavati. 2012. Evaluation of mobile and pervasive speech applications. *Speech in Mobile and Pervasive Environments* (2012), 219–262. DOI: https://doi.org/10.1002/9781119961710.ch8

[34] Keith Vertanen and Per Ola Kristensson. 2009. Automatic selection of recognition errors by respeaking the intended text. In *IEEE Workshop on Automatic Speech Recognition & Understanding (ASRU'09)*. IEEE, 130–135. DOI: https://doi.org/10.1109/ASRU.2009.5373347

[35] Keith Vertanen and Per Ola Kristensson. 2010. Getting it right the second time: Recognition of spoken corrections. In *Proceedings of the Spoken Language Technology Workshop*. IEEE, 289–294. DOI: https://doi.org/10.1109/SLT.2010.5700866

[36] 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*. ACM, 566–573. DOI: https://doi.org/10.1145/3027063.3027066

[37] Nicole Yankelovich and Jennifer Lai. 1999. Designing speech user interfaces. In *Proceedings of the CHI'99 Extended Abstracts on Human Factors in Computing Systems*. ACM, 124–125. DOI: https://doi.org/10.1145/632716.632793

[38] Chengxiang Zhai and John Lafferty. 2017. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the ACM SIGIR Forum*, Vol. 51. ACM, New York, NY, 268–276. DOI: https://doi.org/10.1145/3130348.3130377

[39] Shengdong Zhao, Pierre Dragicevic, Mark Chignell, Ravin Balakrishnan, and Patrick Baudisch. 2007. Earpod: Eyes-free menu selection using touch input and reactive audio feedback. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1395–1404. DOI: https://doi.org/10.1145/1240624.1240836

[40] Yu Zhong, T. V. Raman, Casey Burkhardt, Fadi Biadsy, and Jeffrey P. Bigham. 2014. JustSpeak: Enabling universal voice control on Android. In *Proceedings of the 11th Web for All Conference*. ACM, 1–4. DOI: https://doi.org/10.1145/2596695.2596720

[41] Li Zhou, Suzanne V. Blackley, Leigh Kowalski, Raymond Doan, Warren W. Acker, Adam B. Landman, Evgeni Kontrient, David Mack, Marie Meteer, David W. Bates, and Foster R. Goss. 2018. Analysis of errors in dictated clinical documents assisted by speech recognition software and professional transcriptionists. *JAMA Network Open* 1, 3 (2018), e180530–e180530. DOI: https://doi.org/10.1001/jamanetworkopen.2018.0530

Received April 2019; revised March 2020; accepted March 2020