EDITE: Voice-based mobile text editing

Beatriz Portugal
Técnico Lisboa
biamnportugal@gmail.com

ABSTRACT
The reduced size of virtual keyboards and the lack of tactile clues make them difficult to use, error-prone and slow. This is particularly relevant for blind users. The use of voice interfaces for mobile text entry and editing has proved to be a viable alternative. However, despite its advantages, blind users face challenges revising the text and spend 80% of their time editing, being forced to use the virtual keyboard [3]. In this dissertation, we propose Edite, a voice-based system that allows users to edit text in a simple and fast way, with speech recognition error recovery. Three formative studies were conducted that allowed us to deduce the requirements for a voice editing system. We infer that operations such as read, insert, delete, replace, and redo the last operation are desired by users and that both speech recognition error recovery and clues to system state would improve users' experience. In order to evaluate our solution's usability, we conducted a user study with 8 screen reader users, in which we compared our solution to the traditional text editing method provided by Android's Talkback. Results show that Edite is faster than Talkback and users prefer voice interaction to review and correct text. Finally, implications for the design are suggested to improve mobile text editing voice interfaces.

Author Keywords
Visual impairment; Mobile devices; Text editing; Voice-based interaction; Mobile accessibility; Error Recovery

1. INTRODUCTION

In the past decade, mobile touchscreen devices such as smartphones have been widely adopted by blind people thanks to screen readers like VoiceOver[1] and Talkback[2]. A blind user can explore and select items on the screen as the screen reader tells the action result, allowing efficient use of applications. Alongside smartphone adoption, speech interface has been widely used, mainly because of the launch of virtual assistants like Siri, Google Now, Cortana and Alexa. The integration of speech recognition in mobile devices allowed users to perform tasks that do not require visual feedback, in a more natural manner. Blind users tend to prefer this interaction modality mainly in text entry tasks, which, performed through a virtual keyboard, although using a screen reader, becomes time-consuming and error-prone. Azenkot et. al. [3] explored speech input by blind people on mobile devices and found that speech was nearly 5 times as fast as the keyboard. Despite the advantages of using speech for text input, people face challenges when reviewing and editing a speech recognizer's output, often resorting to using the keyboard, spending 80.3% of their time editing.

Reviewing text can be challenging because users depend on audio feedback to do it and the system is likely to output words that sound like the user input but are incorrect. To edit text, there are some complex tasks like cursor navigation, access to context menus to perform clipboard and selection operations and adjust the navigation granularity (character, word, line), that depend on gestures that can be hard to perform. To our knowledge, no one has explored speech as a non-visual text editing modality, as this is still an accessibility problem. In order to improve user experience in this context, we propose Edite, a voice-based system that allows users to edit text in a simple and fast manner, with speech recognition errors' recovery. We explored relevant and desirable functionalities through three formative studies. The results led us to the design guidelines and user requirements to implement Edite. To evaluate our solution, we conducted a user study with 8 people that depend on screen readers to use mobile touch screen devices. In this study, we compared Edite to Android's Talkback method with the QWERTY keyboard. Results show that Edite is faster than Talkback and users prefer this voice-based interaction to review and correct text. Our main contribution is the characterization of the current nonvisual text editing problems, analysis of voice interaction by blind people, elicitation study on how blind people would edit text by voice and the results of the comparison between Edite's novel interface and Talkback. We finish with implications for the design of future mobile, voice-based, text editing interface.

2. RELATED WORK

In this section, we discuss previous work in four topics: we analyze related research on text entry, text editing and use of voice user interfaces for blind people, as well as metrics used to evaluate text entry and editing performance. Finally, we describe the existing gap in non-visual techniques.

2.1. Text Entry for Blind Users

Currently, most of the mobile devices have accessibility services, like Apple's VoiceOver and Android's Talkback, to allow blind users to perform several tasks like text input. Users can explore the keyboard by dragging their finger and have the

[1]https://support.apple.com/pt-pt/HT204783
[2]https://support.google.com/accessibility/android/

keys to read aloud as they touch them. While sighted users see the keyboard before pressing the pretended key, blind users, apart from not having physical references of the keys, rely on their mental model of the keyboard and the size of the device. Insertion of punctuation marks, special characters and numbers is also a challenge, as it requires keyboard change. In order to reduce the number of targets in mobile text entry, some solutions have been proposed. Braille-based keyboard favors users in the target location and precision because of the reduced number of keys. A traditional Braille keyboard consists of 7 main keys paired with each of the six dots of the Braille code and a space key. In solutions like Perkinput[4], BrailleTouch[5], and HoliBraille[9], the system generate a braille keyboard on the screen and the user places one finger in each dot. Trindade et. Al.[11] takes into account the user preference for physical buttons, and proposed Hybrid-Brailler, an input solution that combines physical and gestural interaction to provide fast and accurate Braille input.

Regarding the voice input, mobile device users can dictate text in any text entry scenario on iOS and Android devices, as automatic speech recognition is well integrated on mobile platforms.

2.2. Text Editing for Blind Users

Nonvisual mobile text editing has few contributions in the scientific community. Accessibility services like VoiceOver and Talkback allow users to perform editing operations like cursor movement, text selection, and clipboard operations (i.e. copy, cut, and paste) through contextual menus and gestures. With these menus, "rotor" in VoiceOver and inverted "L" gesture in Talkback, users can define navigation granularity (character, word, line), enable selection operations and perform typographical operations. To move the cursor, users can use the volume keys or perform vertical (iOS) or horizontal(Android) swipes.

Although these solutions enable users to perform any editing tasks, they depend on complex gestures, leading users to delete all the entered text and start over instead of reviewing and correcting the text.

With the braille keyboard on the back of the device, Hybrid-brailler[11] uses the touchscreen for gestural interaction for editing operations. Editing operations start by pressing the edit mode key. After that, the user can move the caret with horizontal gestures, select text by holding one of the thumbs on the screen and performing caret movements and perform clipboard operations by holding the selection thumb and performing a vertical gesture.

Some braille-based solutions like BrailleType[10] and BrailleTouch[5] offer a delete option for the last inserted chord.

EDITalk[7] is an eyes-free word processing interface that enables users to revise the text by using Text-to-speech and to perform actions like highlight, comment, insert, delete and replace without depending on cursor movement. The system can read all the contents in the text field and provides feedback for operations performed by the user.

Dragon[3] solutions offer editing, formatting and text correction methods just by using voice. Most of the editing operations, such as text deletion or replacement, text formatting, and annotations require visual feedback, making them unsuitable for eyes-free use. None of these two solutions were designed for visually impaired users.

2.3. Voice user Interfaces

Speech has been widely used in applications and tools like voice-activated personal assistants, like Siri and Google Now and mobile devices navigation tools. Abdolrahmani et. Al.[1] says that these virtual assistants are used mainly for tasks like creating alerts/setting reminders, searching for information and checking the weather. This can reduce the time and effort needed to access applications and does not require the target location on the screen, improving the blind users' experience. Speech interaction has the disadvantage of misinterpreting commands in noisy environments. Besides personal assistants, there are navigation and control solutions like CaptiSpeak[2] and JustSpeak[12]. JustSpeak is a voice control solution for non-visual access to the Android operating system that enables system-wide voice control to any application and provides more efficient and natural interaction with support of multiple voice commands in the same utterance. CaptiSpeak is a voice-enabled screen reader that is able to recognize commands like "click <name> link," and "find <name> button,".

2.4. Evaluation Metrics

Regarding text entry efficiency metrics, Word Per Minute is the most used metric. User actions are not considered in this measure, only the number of words in the transcribed sentence as well as the time the user spent on it. Some researchers use other metrics like Characters Per Second, Keystrokes per second and Gestures per second. To measure the error rate, usually, researchers use the Minimum String Distance Method method. The algorithm yields the minimum distance between two strings defined in terms of editing primitives. The primitives are insertion, deletion, and substitution. The idea is to find the smallest set of primitives that applied to one string (transcribed text) produces the other (presented text). The number of primitives in the set is the minimum string distance (MSD). Regarding speech as a text entry modality, researchers consider metrics that distinguish between errors made by the user when pronouncing incorrect words and errors by the speech recognizer.

Concerning text editing, Fuccella et. Al.[6] proposed a set of phrases and editing tasks that cover all the editing operations (cursor movement, text selection, and clipboard operations). Trindade et. Al. [11] used a similar set of phrases that would require several editing operations. During the initial sentence with errors were read to the participant and he was asked to correct the phrase. Although this procedure guarantees internal validity, Trindade et. Al.[11] says that performance and overall error awareness may be different when users edit their own sentences. To evaluate text editing tasks, researches measure the task time, task success rate, number of edit events, editing ratio, calculated using the Minimum String Distance

[3] https://www.nuance.com/dragon.html

(MSD) measure. This measure indicates how far participants are from finishing the editing tasks considering the original sentences.

Usability text entry and editing methods can also be measured through subjective user impressions, using a SUS test (System Usability Scale).

Nonvisual text entry has been actively explored in the last decade and we can find efficient modalities like speech, a popular alternative to the on-screen keyboard with VoiceOver. Yet, nonvisual text editing is still a difficult and complex task and users tend to avoid editing operations. To our knowledge, no one has explored speech as a text editing modality for blind users.

3. PERCEPTION OF VOICE-BASED INTERACTION PROCESS

3.1. Observations

We performed 15 observations sessions, over 18 months, on the challenges blind people face when interacting with touchscreen mobile devices. All participants were visually impaired and their experience with mobile devices with the touchscreen was variable. The main results regarding text manipulation and voice interaction are as follows:

Text entry with QWERTY keyboard. Novice users find this technique complex. Keyboard keys are small and have no physical cues. This modality requires a long exploration until the desired key is found and memorizing the position of the characters in the keyboard. For expert users, this is an efficient solution. Still, it is not faster than voice input.

Voice input. To perform voice text input, Android's and iOS's keyboards have a dictate button. Although this is a fast way to input text, the reduced size of the dictate button, usually smaller than the other keys, slows down the task.

Voice Interaction. During the sessions, many participants showed interest in virtual assistants and text dictation. Despite this, users had difficulty understanding when speech recognition ends when entering text, which led to the insertion of unwanted text.

Text Revision. Most participants did not revise the text. Those who did it tended to hear the word as soon as they hit the space key. In most cases, the resulting text turned out to have errors that went unnoticed, as extra characters in words.

Text editing. The most commonly used technique for editing text is to use the backspace key to delete text until they find the error and rewrite. In most cases, the error is detected as soon as the character or word is typed, and users then proceed to edition. Users rarely resort to clipboard operations and most of them do not know much about its existence.

Character navigation. Character navigation is not a popular technique given the way users review and correct text. Still, some users use the volume keys to navigate the text. The biggest challenge when navigating through the text is that users are not aware of the cursor position regarding the output word.

Access to the context menus. Revision, navigation and editing tasks became more efficient when using Android's and iOS' context menus. Most of the participants did not use these menus because of the complex gestures they had to make to access them.

3.2. Workshop "Speak with your mobile phone"

To better understand how blind users interact with mobile devices just by using their voice, their needs and challenges in this context, we conducted a workshop: "Speak with your mobile phone".

3.2.1. Participants

We recruited 12 blind participants (six males, six females). All of them were smartphone users except for participant P10 that had a physical keyboard phone. We needed 4 collaborators in order to guide the workshop in 4 groups of 3 people.

3.2.2. Procedure Apparatus

We used 8 Android mobile phones with voice-activated Google Assistant installed. Concurrent use with Talkback is limited as the two solutions provide simultaneous audio feedback, making it impossible to perceive virtual assistant feedback. Therefore, Talkback was turned off during the session. The session was about 2 hours long. At the beginning of the sessions, we asked participants how they perform text entry and editing tasks and how often they use speech as input. We then explained what are virtual assistants and how to interact regarding dictation, voice volume and command composition. Then, we configured voice recognition in each mobile device. After that, in each group, participants were asked to perform a set of representative daily tasks: Ask the time; Check the weather for the day after; Set an alarm; Set a reminder; Make a call; Write an SMS; Search something on internet.

Every participant in each group had a chance to perform each task as many times as they wanted. Collaborators gave suggestions on how they could interact with the virtual assistants and participants were encouraged to try other commands. At the end of the session, we showed the limitations of this kind of virtual assistants and we helped them install the Google Assistant on their mobile phone if they wanted. We recorded the session audio, with users' consent, and then we deduced the success rate and the challenges participants faced during the session.

3.2.3. Results and discussion

Participants achieved a success rate of 96,43%. After analyzing the session and taking into account participants considerations while performing the tasks, we present the system advantages that led to efficient voice interaction:

Command versatility. Participants used different types of commands and expressions and the system was able to recognize different types of commands to perform the same action like "I want to know the weather for tomorrow", "What is the weather forecast for tomorrow?", "Is it going to rain tomorrow?"

Audio feedback with earcons. The system play sounds depending on the system state to let the participant know what kind of action they can do. For example, when participants

said the wake phrase “Ok Google”, the system played an earcon to let them know it is listening. When participants finished the command, a new earcon was played to show the voice caption is over and the command is being processed.

Continuity of speech. The system allows interactivity when it needs confirmation from users using questions like “Do you want to send or change?” “Do you want to save the reminder?”

Filtering cordial expressions and speech errors. The system supports the use of cordial expressions in the command formulation like “please” and word repetitions like ”What is the weather forecast for for tomorrow?"

On the other hand, users faced some challenges when using the system, described as follow:

Unnatural system activation. A lot of participants used to forget to say the wake phrase “Ok Google” before the command. In these cases, the collaborator asked them to repeat the command using the wake phrase.

Inconsistent feedback. Users got confused with the different feedback system gave in response to the same actions or commands.

Enlightening feedback. In some cases, the system feedback was not enough for users to perceive the action performed. For example, when they set alarms for 9 o’clock, the system responded with an “alarm set” without specifying the exact time.

Wake phrase repetition triggers unwanted action. In some cases, the system did not respond immediately to the wake phrase, leading participants to repeat “Ok Google” until they ear the feedback. In these cases, the system triggered a search on google about “Ok Google”. With visual feedback, this does not happen because the system informs visually that voice caption is starting.

Lack of feedback to unknown commands. when users tried to perform commands the system does not provide, the command was used to make a google search without any audio feedback.

Inefficient speech continuity. When the interaction required more than one command due to the lack of information given by the participant, the system did not always deal with responses efficiently, leading to the failure to perform the action. The interaction ended with no audio feedback, presenting a time out message on the screen.

Guaranteed efficiency only for tasks related to mobile phone applications. In case tasks required the use of non-mobile applications such as Internet searches on close pharmacies or restaurants, the feedback was not enough.

Commands and interaction limitations. Although there is speech flexibility in the supported command set, the system does not handle complex commands like “Awaken me tomorrow at 9 am”,” Tomorrow I want to be awake at 9 am” or” What is the weather forecast for tomorrow at lunchtime?". In addition, the system does not support user interaction with the feedback it presents. For example, given this system feedback “Tomorrow the maximum temperature is 20 degrees ”, there is no response from the system to questions such as“ What time less than 20 degrees? ”

Voice caption finished too early. The system assumes commands as completed before the user finished the request.

Diction. One of the participants had some diction problems and the system did not understand most of her commands in the first attempts. It was necessary that the collaborator often asked her to speak slowly. This situation led to some frustration and discomfort in the use of voice interfaces.

3.3. Elicitation Study

After understanding how blind people interact with mobile devices using their voice, we wanted to study what are their strategies to recover from speech text errors by voice, what kind of command they associate with editing actions and how they deal with bad speech recognition.

3.3.1. Participants

For this study, we recruited 6 participants (3 male, 3 female) with visual impairments. All of them used to write on the computer, but only 4 use smartphones to write.

3.3.2. Procedure and Apparatus

For this study, we needed a smartphone and a computer. The smartphone was used for the user interaction with a device, through an app that contained only a button that covered the entire screen. They had to tap the button to start and end the voice caption. Each time the button was clicked, the system played a different sound depending on whether the sound capture was being turned on or turned off. With the computer, we simulated all the audio feedback to every action the user could make. This feedback was generated using ResponsiveVoice JS, a Text-to-speech application. Most of the interactions were previously predicted, so we had ready to use feedback on the application.

At the beginning of the session, we explained to participants that it is possible to interact with mobile devices by voice and how this interaction should be performed. After that, we asked them some information about how they enter and edit text and if they perform any task by voice. We, then, explained that in this study we would simulate text entry by voice and error correction through 11 tasks. In each task, users were asked to dictate a specific sentence and then we introduced errors in the sentence that was transcribed, with their knowledge. Then, they had to correct the sentences with the commands they consider suitable in each situation. We only asked them to use commands simple and concrete. We chose tasks that would require simple actions, like delete a word or a character; actions with text references, like delete a repeated word or replace the first occurrence of a word; navigation actions, introducing non-audible errors, users had to navigate through the characters; and simulation of bad speech recognition, by not recognizing one specific word, forcing users to find other correcting strategies. We started with test tasks with no errors to simulate all the procedures: listen to the sentence they had to dictate, start voice capture, dictate the sentence, end

voice capture, listening to the transcribed sentence. Then users would have to find the errors and correct them.

We record the session audio, with participants' consent, to deduce the task success rate. We also observed the challenges participants faced during the session for further analysis.

3.3.3. Results and discussion

Some participants struggled to find the error in the sentence or decide what strategy they should adopt to correct the error. In these cases, we suggested strategies and encouraged them to finish the task, considering the task incomplete. Regarding error detection, they only found it difficult to detect the non-audible error, where we suggested them to read character by character. When they asked for help in error correction, we suggested them to perform operations on a character level, to tell the system where to write the text and to adopt a new correcting strategy other than repeating an imperceptible word.

Said that participants achieved a task success rate of 89,4%.

Command structure. Concerning commands formulation, in every request, they used a verb followed by an expression. Some participants used courtesy expressions like "please".

Verbs: Participants used several verbs to refer to the same action. These verbs were used in infinitive mode or imperative mode. When they wanted to insert text, users sometimes omitted the verb, saying only the text they wanted to insert.

Expressions: Depending on the tasks, users chose to perform actions on different granularities (characters, word, line, phrase). Most of the commands used a transcription of the text to refer the what they wanted to edit. In some cases, the transcription was preceded by a granularity (like "word fly"). In some cases, participants mentioned the expression position in the text using words like first, second and last followed by the granularity, like "first word" or "last character". Sometimes, the expression mentioned above was used with position references to other text portions with words like after and before. Users also mentioned where to write the text with expressions like "at the beginning" and "at the end".

Command segmentation: Participants tended to combine commands above described like "command and command" or "command and after command". Besides this structure, some participants added courtesy expressions on the end of the commands or showed their intentions to perform actions with expressions like "I want. . . "

Correcting strategies. Overall users resort to four types of commands: insert, delete, replace and read. They combine all these operations to perform all the required actions. Regarding the bad recognition simulation task, those who completed the task started by trying to replace the word and then, after noticing that it is being misunderstood, they chose to spell the correct word.

Observed behaviors. At the beginning of the session, participants were reluctant that the system did not understand their requests. Throughout the session, they got more confident and tried new commands. We observed some other interesting users' behaviors when performing the tasks, described as follows.

Using the voice activation button: At the beginning of each task, participants tended to press the button to start voice caption so that they could utter the command. In the following statements, the use was not so efficient. Participants P3 and P5 always forgot to tap the button at the end of the utterance. Participants P2 and P6 simply uttered the commands without pressing the button. For the remaining participants, the sound for starting and ending voice caption made it easy to understand when they could make the requests.

System intelligence expectation: Participants P3 and P4 showed that they did not trust the system, questioning if it would understand their commands. By contrast, participant P2 asked the device to correct the error instead of giving detailed instructions.

Precision assumption: We asked participants to always review the final sentence. Participant P6, in most cases, assumed that the command had been well interpreted and that the action had been successfully performed, by not confirming the final result. In some of these occurrences, the sentence still contained errors.

Human perception and dialog expectation: While most participants uttered commands independent of each other, participant P5 tended to assume continuity of speech. Her interaction contained expressions like "again", assuming the system remembered her last actions. This participant also used courtesy expressions in the middle and at the end of commands.

Assumption of cursor existence: Throughout all sessions, there were only two references to the cursor, from participant P3. He requested the cursor to be placed at a certain location and, after that, uttered commands referring to the text and not the cursor.

3.4. Dragon Evaluation

Dragon Naturally Speaking is an intelligent speech recognition solution that allows the creation and edition of documents using speech as a modality of interaction.

The major disadvantage of this solution, in the context of this work, is that it was not designed for blind people, but rather to assist daily tasks in a faster and more effective way. The solution does not provide audio feedback, demanding visual search of the text to be edited, for button identification and for system state perception.

The solution is limited to some idioms: English (US), English (UK), Canadian English, and German.

In order to explore the solution and the feasibility of combining it with an accessibility service, we tested the mobile version of this product, Dragon Anywhere. This application presents a main dictation screen with a text field, an action bar where it is possible to perform document management actions and a menu to access documents, auto-texts lists, word lists, settings, and help. For dictation or command execution, it is necessary to activate the voice caption with a tap on a microphone button. After that, the system only provides visual feedback, changing

the button color. No sound is played to show if the system is listening or ended the capture.

All the text and commands have to be uttered in the idiom defined. Before every command that involves a text portion, this has to be selected through the command “select –text-“. After that, it is possible to perform actions like “delete that”, “bold that”, “Copy that” and “correct that”.

After the text is selected, if the user does not say the command exactly as expected, the selected text is deleted and the command is entered. For example, if all text is selected and the user says “ delete” instead of “delete that”, the text is deleted and the word is written. If the command is uttered without selecting text, it is also entered in the text field as speech input. Always having to select text doubles the number of commands, increasing the solution complexity. We tried to use this application with Android’s accessibility service, Talkback.

When the text field is selected, the screen reader outputs the content in the text field. Not all buttons displayed on the screen are labeled, such as the microphone, making it impossible to recognize its functions. Talkback informs the user of some of the changes in text. If this happens and the microphone is active, the Talkback’s feedback is perceived as an input, and an infinite cycle of capture and insertion begins. Using earphones, this situation does not happen.

Besides the lack of audio feedback, some commands open windows without user knowledge and the Talkback’s feedback does not provide any information about the cursor location, making it impossible for a blind user to perceive this information.

3.5. Design Guidelines

Through the formative studies conducted, we saw that users tend to prefer speech voice input and find it easier to perform. Regarding text editing, they struggle when using the context menus provided by Android and iOS. In most of the cases, they just use the backspace key to erase all and start over. Concerning the voice interaction, it is essential to provide a complete and efficient audio feedback of all performed actions and system state. The wake phrases are not effective because of diction problems. The disadvantage of button activation is the target location. Users do not understand when voice caption is over and end up inserting unwanted text. Taking all the results and user behaviors into account, we defined the design guidelines for our solution, described as follow:

- Allow reading the text field content in a simple way, enabling error perception;
- Voice capture should be activated in an easy and fast way and should be turned off when the user finishes the utterance;
- All actions should generate audio feedback;
- It should be possible to perform actions on every text granularity (character, words, phrase, all text).
- The system should support command conjunction;
- Undo the last action performed;
- The system should be semantically intelligent;
- It should be possible to refer text by direct reference or transcription;
- Allow use of text position references to refer expressions;
- It should be possible to refer place of text editing or text insertion without using a cursor;
- It should be possible to recover from error through user-system interaction;
- During interaction, minimize chances of speech recognition errors by asking yes or no questions;
- Handle with incomplete commands;

4. EDITE: A VOICE-BASED TEXT EDITING SYSTEM

In order to provide the blind community with a fast and easy way to edit text in mobile devices, with a particular focus on the difficulties and needs we observed in the previous referred studies, we propose Edite, a voice-based text editing system with error prevention and recovery.

4.1. Solution Conceptualization

4.1.1. System design

The purpose of the system is to make the execution of the editing tasks easy and fast. That said, dictation and editing command are performed through voice commands. The system has the ability to end the voice caption when the user finishes the utterance, as well as the user. The command is processed, the action performed and the system returns the result of the action via audio.

4.1.2. Feedback

Complete audio feedback is essential for the efficient use of a system in this context. Edite provides audio feedback after all commands, telling the results of the triggered actions or reading text field content if required.

Based on Audio Signifiers for Voice Interaction [4] article, we included Voice-Interaction Signifier to help users understand what kind of requests they can make and when. We use Explicit verbal signifiers to recover from incomplete commands by asking questions that lead users to give the information system needs. Implicit verbal cues are used when the system is speaking and the user shows the intention to request something. In this case, the system stops speaking to let the user know that it is listening. We also adopted two Nonverbal sounds or earcons. These are used when the voice caption is turned on and turned off to let the user know that the system is ready to receive commands.

[4] https://www.nngroup.com/articles/audio-signifiers-voice-interaction/

4.1.3. Command structure:

According to the actions used in the elicitation study, the input and text editing actions were grouped into a set of five commands through which users will be able to perform all actions: insert, delete, replace, read and undo. Users can read the text in three different ways: read, read word by word, with pauses between each word, and spell. The general structure of a command requires an action (insert, delete, replace, read , read words, spell) and an argument, except for "read words" that does not require any argument. "read" and "spell" command can also be invoked without arguments and the "insert" action can be omitted for text input. The system also accepts a set of commands in the same utterance, up to two commands.

Regarding the arguments, this can be inputs, for insertions or replacements, or a combination of the following: a transcription, a granularity (character, word, phrase, text) or the position the expression occupies in the text, through expressions like first, second and last. These expressions can also be combined with references to an absolute position in the text or in relation to other expressions.

4.1.4. Error recovering:

In order to make our solution semantically intelligent, regarding the command construction, we allow the use of a limited set of synonyms for each action, expression, or reference. This set was based on the commands used by the participants in the elicitation study and aims to give the user flexibility and enable a natural discourse.

One of the biggest problems associated with voice interfaces are speech recognition errors. Some expressions are replaced by others with a similar sound and go unnoticed. This system provides a speech recognition error recovering functionality and can also recover from commands not anticipated by the system. This process is done through three recovering levels and user interaction. At each level, the system tries to infer a new command and asks if the user wants it run. All questions require yes or no answers in order to minimize the chance of new speech recognition errors. If the system does not find a match in a specific level, move to the next level. If the user answer is "no", the system move to the next level. If there is a match and the user answer is "yes", the system process and executes the new command. The recovering levels are as follow:

Command repetition. In order to make the experience more natural and closer to speech, in the first level of recovery the system prompts the user to repeat the command. the user will have the chance to say it again more carefully, as in the study of elicitation, which will allow the perception of other words by the speech recognizer. This way, the new command could correspond to the set command possibilities offered by the system.

Verification of the most confidently recognized words. With the speech recognizer used, it must be possible to consult the list of recognized words with greater confidence in the utterance. At this level of recovery, various combinations of new commands are made with the words returned by the speech recognizer and check if there are matches with the possible commands.

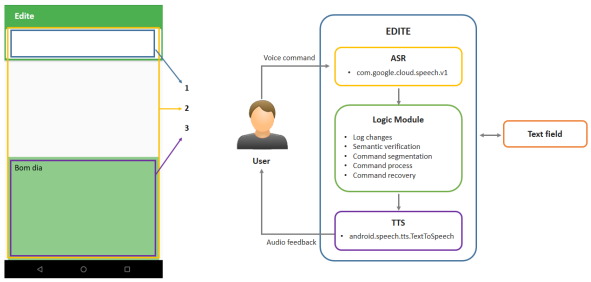

Figure 1. Edite interface (left); Edite architecture (right)

Edit distance verification. In this level, the command fields are compared with the possible options through the Levenshtein algorithm, which calculates the similarity between text portions based on the number of characters inserted, removed or replaced.

If the first word (action) has no match, the utterance is perceived as an input. If the first word corresponds to an action, the system will check the arguments according to the three levels mentioned and if there is no match, the system outputs a message of unrecognized command.

If an incomplete command is issued, the system interacts with the user in order to complete the command.

4.2. Implementation

4.2.1. Architecture

This system was implemented as an Android application. Solution implementation is based on three functional modules (Figure 1):

Automatic Speech Recognition (ASR). This module allows the reception of voice commands and dictated text. the solution does not include natural speech processing, so we used Google's text-to-speech recognizer. Google Cloud Platform provides an API that allows speech recognizer access, we used the com.google.cloud.speech.v1 version. All the content received in this module is sent to the logic module, which processes the information and triggers actions.

Logic Module. It this module, all the utterances recognized by the ASR module are processed according to the solution grammar. It is also in this module that is stored a record of all text changes and performed actions, allowing the execution of redo command, and the set of synonyms supported in the commands.

Text-To-Speech (TTS). This module allows the vocalization of all the feedback provided by the application. It allows the user to read and review the text on the screen and give feedback on all actions performed. In case ASR is enabled, TTS stops immediately and the system starts listening to the user commands. We used the TTS API provided by Android, android.speech.tts.texttospeech.

4.2.2. Interface.
Our system contains two text fields. One of them is located in the lower half of the screen (Figure 1, marked with 3), where all the inserted text is visible; the other shows the text being inserted, at the top of the screen (Figure 1 marked with 1). The only application button has the function of activating and deactivating voice recognition and covers about 90% of the screen (Figure 1 marked with 2).

4.2.3. Speech capturing function.
Speech capturing is activated with a tap on the only existing button. After that, the systems play an earcon, to let the user know they can speak. the user finishes the utterance, the system detects the absence of speech and ends the capture immediately. The user also has the possibility to finish the capture with another tap on the button. When the voice capture ends, the system plays another earcon to inform the user.

4.2.4. Command processing and feedback.
The utterance analysis and correspondence with the possible commands were mostly done through manipulation and comparison of strings. Initially, we check the number of existing commands. After that, we check if the first word of the command is an action. After, we process the arguments, if any, to locate the text portion to be changed. The system does all the possible command combinations, according to grammar. If there is a match, the command is executed, if not, the system starts the recovery mode. Regarding the recovering mode, from speech recognition errors, we implemented command repetition and edit distance verification, as well as incomplete commands recovery.

5. USER TESTS

The usability of the system proposed by us was compared with the most used method for blind text input and editing, the QWERTY keyboard using an accessibility service. In this case, the performance was tested on a mobile phone with an Android operating system using Talkback. Following we described the study conducted and the results.

5.1. Participants

We recruited 8 participants (4 male, 4 female), from Fundação Raquel e Martin Sain, that depend on screen readers to use mobile touchscreen devices. We required participants who have experience in using smartphones, who could use screen readers and who could write on a QWERTY keyboard. All of them are used to write and revise text, after entering each word, but only two of them (P6 and P8) perform all editing operations. The remaining participants use the backspace key to erase and rewrite.

5.2. Procedure and Apparatus

We used a Nexus 5X to perform all the tasks in our solution and with qwerty keyboard and Talkback. We used AZ Screen Recorder to record the screen for future analysis.

The sessions took place in a quiet room and had a duration of 45 minutes. We started by telling users that we were going to compare two solutions to write and edit text through a set of seven tasks for each solution. Tasks were designed so that the participant had to correct sentences with speech recognition errors, previously written on the screen. The errors introduced in sentences simulated the following scenarios, some of them referenced by Hong et. al. [8]:

- T1: speech recognizer ends the voice capture before the end of the utterance;
- T2: extraneous words or sounds are captured
- T3: user says a semantically wrong word in the context of the sentence, corrects it, and both are inserted;
- T4: user utters a word, speech recognizer perceives a word with a similar sound
- T5: user dictates a word, speech recognizer perceives two words with similar sound
- T6: user says two words, speech recognizer perceives a word with a similar sound
- T7: user says a word set, speech recognizer perceives a new word set similar sound

For each condition, we trained the participant on how to use that specific solution and then participants were asked to perform the task set. For Talkback, we taught them how to read the text, the writing technique and how to navigate. Regarding Edite solution, we explained how to turn the microphone on and off, what kind of commands they could do and that the system could ask questions in order to recover from errors. For practice, we suggested tasks that involved all the possible actions in the two conditions.

Before each task, the type of error entered was mentioned, without specifying the error or the location. Then the user was told the correct sentence and they had to correct it using the techniques we taught for each condition. The task time started as soon as the participant touched the screen, or any other key, for the first time.

At the end of the session, we asked participants to rate on a Likert scale from 0 (nothing) to 7 (very) how fast, easy and useful they found both solutions. They were also asked to say which of the solutions they prefer to enter text and edit text.

5.3. Dependent Measures

From the screen and session records, with participants' consent, we collected task time, number of edit events, counting each command for Edite and each touch or key press for Talkback and edit distance between the correct sentence and the final sentence. We also collected qualitative measures like users feedback.

5.4. Design and Analysis

We performed a within-subjects so, to avoid learning between conditions, we had two versions of each task. The order of the testing conditions was random, as well as the order of the tasks. Overall we had 8 participants x 2 conditions x 7 tasks = 112 sentences.

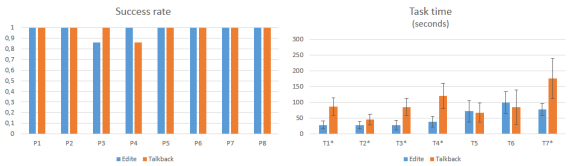

Figure 2. Success rate results for each participant (left); Task time in seconds (right); significantly Edite interface (left); Edite architecture (right); statistically significant values are market with *

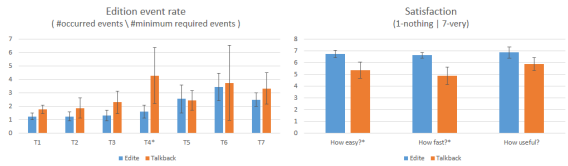

Figure 3. Edition event reate for each task (left); Satisfaction results (right); significantly Edite interface (left); Edite architecture (right); statistically significant values are market with *

We performed Shapiro-Wilk tests on all dependent measures. For normally distributed values(Shapiro-Wilk p<0.05), we applied a Paired T-Test. For the measures that were not normally distributed, we applied the Wilcoxon test.

6. RESULTS AND DISCUSSION

In this section, we show the comparison between Edite and Talkback regarding success rate, task time, edition event rate and satisfaction. We then perform a detailed analysis of Edite's performance.

6.1. Success rate

Overall, success rate (2 was high for both solutions, with the same values of mean and standard deviation ($\bar{x} = 98,25\%, \sigma$=0,0463). Participants P3 and P4 didn't achieve a 100% success rate due to incomplete completion of task T7 in Edite and Talkback, respectively. In both cases, the final sentence had a distance of 3 from the correct phrase.

6.2. Task time

The task time (2) was measured from the participant's first tap on the screen until the task ended. This measure is expressed in seconds. As for the time analysis of each task tested, participants were significantly faster in performing tasks T5 and T7 using Edite.

The only task that took longer to perform on the Edite system than with Talkback was task T6, due to the occurrence of speech recognition errors during sentence correction. The same speech recognition errors that the user was trying to correct. Even so, the tasks were completed using only the voice as a modality.

6.3. Edition event rate

We deduce edition event rate by combining the number of editing events that participants held in each task in each solution with the minimum number of events required through the metric :

$$\frac{\#occurredevents}{\#minimumrequiredevents}$$

The result of this metric (Figure 3) represents the relationship between the number of events performed and the number of necessary events. The optimum value is 1, meaning that the task was performed with the minimum number of events. The higher the value, the greater the difference between the number of events held by the participant and the number of events required to accomplish the task. Edite achieved a significant difference (t(8)=-2,56; p<0,038) only in task T4. We can conclude that the Edite system is more efficient in replacing a single word for another.

In the first tasks, the difference may not have been significant due to the fact that the correction does not require many navigation events when using Talkback. In the last tasks, it could be because of the occurrence of speech recognition errors when trying to correct or insert more than one word. Even so, although there are no significant differences for most tasks, Editing is significantly faster than Talkback.

6.4. Satisfaction

Overall, users consider the Editing system to be the easiest method to use (Z = -2.232, p<0.05), faster (Z = -2.558, p<0.05) and more useful (Z = -1.807, p = 0.071), although this last metric is not statistically significant (Figure 3). When asked which system they preferred to type and edit text, all participants answered Edite for both techniques.

During the session, some participants shared their thoughts. Regarding Talkback, they said that they were "focusing on many things at the same time". Concerning Edite, users shared comments like "I want this on my mobile phone", "this would make life easier and better for a lot of people", "it just takes a tap to write and it's done".

6.5. Edite accuracy evaluation

Throughout the evaluation, excluding training tasks, 223 commands were used, of which 86.55% were well recognized and were expected by the system. From the remaining 13.45%, 7,62% were misunderstood by the speech recognizer and 5,83% were understood but not predicted by the system (5.83%). In both cases, the system tried to recover the commands, which 8,07% were recovered and executed successfully.

The accuracy of Google's voice recognizer used in the Edite system can be measured using the number of well-recognized commands (206) and the total number of commands (223), which results in an accuracy of 92.4%

Regarding Edite's accuracy in the recovering of misunderstood command by the speech recognizer, we analyzed the number of misrecognized commands and those that have been recovered, resulting in a recovering accuracy of 58,8%.

Although both methods achieved high success rates, the speech method has been shown to be 1.8 times faster and also more efficient, taking into account the task time and the rate of editing events. In the tasks where the goal consisted in the

correction, insertion or removal of a single word, the users were 2.7 times faster to perform tasks with the Edite system over Talkback, while tasks that involved correcting word sets with similar sounds, they were 1.3 times faster to complete tasks with Edite.

In addition, it was considered by the participants the easiest and fastest method for editing text.

7. CONCLUSION AND FUTURE WORK

Given the adoption of speech as a text entry modality and the gap in the nonvisual text entry field, we proposed Edite, a speech-based mobile text editing system that allows users to perform editing operations in a simple and fast manner. We compared this solution with the traditional text editing modality, Android's Talkback. Results show that Edite is 1,8 faster than Talkback and users find this solution a fast and easy way to edit text.

The need for improvements are mainly due to misunderstood and unrecovered commands. We would like to increase command flexibility, commands retrieval quality and implement clipboard operations. In addition, and taking into account feedback from participants, we would like the blind community to be given the opportunity to use a system with these characteristics.

ACKNOWLEDGEMENTS

We thank Fundação Raquel and Martin Sain in Lisbon (Portugal) and all participants for their collaboration.

REFERENCES

[1] Ali Abdolrahmani, Ravi Kuber, and Stacy M. Branham. 2018. "Siri Talks at You". (2018), 249–258. DOI: http://dx.doi.org/10.1145/3234695.3236344

[2] Vikas Ashok, Yevgen Borodin, Yury Puzis, and I V Ramakrishnan. 2015. Capti-speak: A Speech-enabled Web Screen Reader. *Proceedings of the 12th Web for All Conference* (2015), 22:1–22:10. DOI: http://dx.doi.org/10.1145/2745555.2746660

[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 - ASSETS '13*. 1–8. DOI: http://dx.doi.org/10.1145/2513383.2513440

[4] Shiri Azenkot, JAcob O. Wobbrock, Sanjana Prasain, and Richard E. Ladner. 2012. Input finger detection for nonvisual touch screen text entry in Perkinput. *Proceedings of Graphics Interface (GI '12)* d (2012), 121–129. DOI:http://dx.doi.org/2305276.2305297

[5] Brian Frey, Caleb Southern, and Mario Romero. 2011. BrailleTouch: Mobile texting for the visually impaired. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 6767 LNCS. 19–25. DOI: http://dx.doi.org/10.1007/978-3-642-21666-4{_}3

[6] V Fuccella, P Isokoski, and B Martin. 2013. Gestures and widgets: performance in text editing on multi-touch capable mobile devices. *Proceedings of CHI 2013* (2013), 2785–2794. DOI: http://dx.doi.org/10.1145/2470654.2481385

[7] Debjyoti Ghosh, Pin Sym Foong, Shengdong Zhao, Di Chen, and Morten Fjeld. 2018. EDITalk: Towards Designing Eyes-free Interactions for Mobile Word Processing. (2018), 1–10. DOI: http://dx.doi.org/10.1145/3173574.3173977

[8] Jonggi Hong and Leah Findlater. 2018. Identifying Speech Input Errors Through Audio-Only Interaction. *Proceedings of the 36th Annual ACM Conference on Human Factors in Computing Systems* (2018), 1–12. DOI:http://dx.doi.org/10.1145/3173574.3174141

[9] Hugo Nicolau, Kyle Montague, Tiago Guerreiro, André Rodrigues, and Vicki L. Hanson. 2015. HoliBraille: multipoint vibrotactile feedback on mobile devices. *Proceedings of the 12th Web for All Conference on - W4A '15* (2015), 1–4. DOI: http://dx.doi.org/10.1145/2745555.2746643

[10] João Oliveira, Tiago Guerreiro, Hugo Nicolau, Joaquim Jorge, and Daniel Gonçalves. 2011. BrailleType: Unleashing Braille over touch screen mobile phones. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 6946 LNCS, PART 1 (2011), 100–107. DOI: http://dx.doi.org/10.1007/978-3-642-23774-4{_}10

[11] Daniel Trindade, André Rodrigues, Tiago Guerreiro, and Hugo Nicolau. 2018. Hybrid-Brailler : Combining Physical and Gestural Interaction for Mobile Braille Input and Editing. (2018).

[12] Yu Zhong, T V Raman, Casey Burkhardt, Fadi Biadsy, and Jeffrey P Bigham. 2014. JustSpeak: Enabling Universal Voice Control on Android. *Proceedings of the 11th Web for All Conference on - W4A '14* (2014), 1–4. DOI:http://dx.doi.org/10.1145/2596695.2596720