GeShort: One-Handed Mobile Text Editing and Formatting with Gestural Shortcuts and a Floating Clipboard

GULNAR RAKHMETULLA, University of California, Merced, United States

YUAN REN, University of California, Merced, United States

AHMED SABBIR ARIF, University of California, Merced, United States
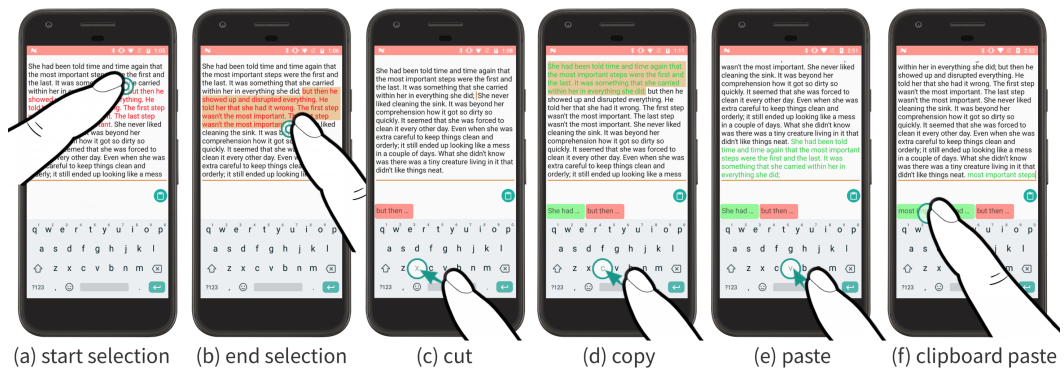
Fig. 1. A cut–copy–paste workflow with GeShort: the user (a) positions the cursor and double-taps to initiate a selection, (b) positions the cursor and double-taps to complete the selection, (c) swipes from the Space to "X" to cut the selected text, (d) swipes from the Space to "C" to copy the selected text, (e) swipes from the Space to "V" to paste the selected text, (f) taps on a previously cut (highlighted in red background) or copied (highlighted in green background) text on the floating clipboard to paste it.

GeShort is a novel method for one-handed text editing and formatting on mobile devices. It uses simple rules to facilitate direct cursor positioning, gestural shortcuts inspired by keyboard hotkeys for editing and formatting, and a floating clipboard to enable delayed, repeated, and block editing. A comparison between GeShort and the default Google keyboard revealed that users perform editing and formatting tasks about 11% and 22% faster, respectively, with GeShort. This is achieved by significantly reducing selection time by 11% and action time by 17%. A second study comparing the clipboard features of the two methods revealed that users perform advanced editing tasks 34% faster with GeShort. Besides, participants find GeShort less onerous in mental demand, physical demand, and effort, which likely contribute to the overall performance gain. They also perceive GeShort as faster and easier to use, feel that its functions are better integrated, thus want to keep using it on their devices.

CCS Concepts: • **Human-centered computing** → **Text input**; **Gestural input**; • **Applied computing** → *Text editing*.

Additional Key Words and Phrases: hotkeys, cut, copy, paste, text formatting, shortcuts, gestural input, touchscreen, virtual keyboard

1 INTRODUCTION

Text entry has become a vital part of our everyday life. Nowadays, we enter text not only on desktop computers but also on the go on our mobile devices. While there is a rich body of work on text entry and error correction on mobile devices, not much work focused on text editing or formatting. However, with the increased use of mobile devices, developing efficient text editing and formatting approaches are the next integral step in the progression of mobile text entry. Besides, text editing requires effective approaches for precise cursor positioning and text selection, which can also benefit error correction. This work defines text *editing* as the process of manipulating existing text with modeless editing operations, including cut, copy, paste, and move [42], not to be confused with *error correction* that involves correcting incorrect text or *revision* that involves rewriting parts of existing text to improve its quality. Text *formatting*, in the context of this work, represents styling existing text with bold, italicized, or underlined typeface.

Both text editing and formatting on mobile devices involve precisely positioning the cursor over the text to be selected, long-tap (500–1,000 ms [15]) to enable the "edit mode", which displays an edit toolbar and two handles to adjust the selection range (see Fig. 2b), adjust the selection range by dragging the handles, then select the intended task from the toolbar. If the intended task is not visible in the toolbar (which is usually the case for formatting tasks), users have to go to a secondary drop-menu by tapping on an icon, then select the option. Precise positioning of the cursor is difficult on smartphones due to the "fat-finger problem" [43]. Prior studies showed that users frequently make mistakes, requiring extra time for correction, when precisely positioning the cursor using touch [8]. The use of a dwell threshold (long-tap) and multiple menu selection actions also add to the time and complexity of text editing and formatting on mobile devices. Performing these actions are even more difficult when holding the device with one hand and interacting with the thumb of the same hand, which is one of the most common postures for mobile interaction [7]. Mobile users frequently use one hand to hold and enter text on mobile devices in situational impairments when the other hand is unavailable, such as when holding a coffee cup or a sandwich with one hand, performing dual tasks (e.g., navigating a desktop browser when texting, etc.), holding the hand of a toddler, or when relaxing on a couch. People who cannot use both hands due to a disability or amputation also use one hand to interact with mobile device. However, we do not include this population in this research.

Further, existing mobile text editing methods do not fully support delayed, repeated, and batch editing, thus not always possible to cut/copy text for use in a later editing episode (delayed pasting), cut/copy chunks of text for pasting in a preferred sequence (repeated pasting), or paste all cut/copied text with a single action (batch pasting). Instead, most methods require users to repeatedly perform the select-cut/copy-reposition-paste action sequence. Recently Gboard, the default Google keyboard [30], introduced a clipboard feature that stores the last five cut/copied text, enabling users to paste those later in a preferred sequence. While this feature supports delayed and repeated pasting (but not batch pasting), accessing the clipboard and locating its content is not straightforward, rather time-consuming and tedious, discussed in Section 3.

We propose GeShort, a method for one-handed text editing and formatting on mobile devices using gestural shortcuts. GeShort facilitates direct cursor positioning by using three simple rules, one-handed text editing and formatting with gestural shortcuts, which are inspired by the commonly

used keyboard hotkeys, and delayed, repeated, and batch editing using a floating clipboard. The remainder of the paper is organized as follows. First, we review the existing works in the area and discuss the basic and advanced editing and formatting features on current mobile systems. We then present GeShort's text selection, editing, and formatting approaches. We evaluate GeShort in two user studies by comparing its basic and advanced features with the default Google keyboard's basic and advanced features. Finally, we conclude the work with reflections on future directions.

2 RELATED WORK

There are not much works focused on text editing or formatting on mobile devices. Fuccella et al. [20] enabled text editing by performing one and two-finger gestures on the screen. In an evaluation, this method yielded 13–24% faster task completion time than Android OS 2.3's editing widget. In a follow-up work, Fuccella and Martin [21] used similar gestures to enable bimanual text editing. This method was 2% faster than Android OS 5.0's editing widget in a user study. Zhang and Wobbrock [47] used similar gestures to enable text editing with one and both hands. In a study, the one-hand approach yielded a 24% faster and the two-hand approach yielded a 17% faster task completion time than Android OS 9.0's editing widget. These methods, however, do not support text movements and use unfamiliar gestures that could be difficult to discover and learn [14]. In general, these works establish gestural interaction as an effective method for manipulating text on mobile devices. These works suggest that multi-finger gestures result in a faster task completion time than one-finger gestures but tend to increase physical efforts.

Ando et al. [4] developed a tap and tilt hybrid method with which users place the cursor at the beginning of the text, hold the "C" key and tilt the device to adjust the selection range, then release the key to copy the selected text. In a follow-up work, Ando et al. [5] replaced tilts with finger slides, where users use the keyboard as a trackpad to adjust the selection range. Both methods were reported to perform better than the Android OS 8.0's editing widget. But the actual performance of the methods are indeterminant due to the use of unconventional evaluation protocol and performance metrics. Besides, these methods do not support common editing tasks, such as cutting, pasting, or moving text. Roth and Turner [36] used the bezel of a device to enable text editing. With this method, users position the cursor at the beginning of the text, initiate a gesture from a specific area of the bezel to indicate cut or copy, then lift the finger at the end of the text to

Table 1. Mobile text editing and formatting methods with their reported performance gains compared to the legacy editing and formatting features of smartphones. Notice that none of these methods support all commonly used editing tasks (i.e., copying, pasting, cutting, moving).

Reference	Interaction	Editing	Formatting	Baseline	Gain	
					Speed	Accuracy
Chen et al. [15]	Bezel gestures	Partial	No	Android OS	30%	−3%
Zhang and Wobbrock [47]	One-hand gestures	Partial	No	Android OS	24%	NA
Fuccella et al. [20]	One-hand gestures	Partial	No	Android OS	13-24%	NA
Zhang and Wobbrock [47]	Two-hand gestures	Partial	No	Android OS	17%	NA
Fuccella and Martin [21]	Two-hand gestures	Partial	No	Android OS	2%	NA
Roth and Turner [36]	Bezel gestures	Partial	No	iOS	−36%	0%
Ando et al. [4]	Tap and tilt	Partial	No	Android OS	NA	NA
Ando et al. [5]	Tap and swipe	Partial	No	Android OS	NA	NA
Gutwin et al. [24]	Two-hand chording	Partial	Yes	NA	NA	NA
Fennedy et al. [19]	Two-hand hotkeys	Partial	Yes	NA	NA	NA
Schweigert et al. [39]	Knuckle gestures	Partial	Yes	NA	NA	NA
Alvina et al. [3]	One-hand gestures	No	Yes	NA	NA	NA

complete the corresponding task. In an evaluation, this method was 36% slower than the Apple iOS 2.0's editing widget. Chen et al. [15] adapted this method to enable cross-application copy-paste, where it yielded a 30% faster copying time than Android OS 4.1.2's editing widget. The study, however, used a mix of 15, 18, 21 sp sized fonts, two of which were larger than the recommended 16 sp [27]. Hence, it is unclear how the method would perform in real-world scenarios. Schweigert et al. [39] proposed using knuckle gestures for text editing tasks. Gutwin et al. [24], in contrast, augmented three push buttons on a smartphone case to enable text editing by performing chords. These methods were not evaluated in user studies. Alvina et al. [3] enabled text formatting by performing gestural shortcuts above the keyboard. Fennedy et al. [19] adapted actual keyboard hotkeys on a virtual keyboard, where users tap on the keys in a sequence or with two thumbs. These methods were also not evaluated in user studies. Findings of these works suggest that exploiting device holding position and posture or external hardware and attachments do not result in a performance gain in short exposures. However, it is unclear whether these methods will improve performance with practice since neither of these were evaluated in longitudinal studies. Table 1 presents existing text editing and formatting methods for mobile devices and their performance gains reported in the literature.

Some have studied cursor positioning with mobile devices. Scheibel et al. [38] developed a virtual (joy)stick controller to control the movement of the cursor. Arif et al. [8] developed a smart cursor positioning system to facilitate error correction. Albanese et al. [1] proposed displaying directional arrows near the cursor to enable adjusting its position by tapping on them. These methods, however, were not compared with standard methods. In a different line of research, Zhao et al. [48] enabled text editing with speech commands. Eady and Girouard [18] developed a deformable prototype to demonstrate cursor control by bending the corners of the device. Le et al. [29] elicited (mostly) back-of-device gestures for the most important shortcuts for smartphones. Darbar et al. [17] used a smartphone as a trackpad, conduit for keyboard shortcuts, air-mouse, and ray casting device to enable cursor positioning in augmented and virtual reality. Pandey et al. [34] developed a predictive system for number editing on smartphones. Some have also developed methods to facilitate error correction on mobile devices [6, 8, 9, 28, 41]. These, however, are outside the scope of this work.

3 DEFAULT TEXT EDITING AND FORMATTING BEHAVIORS

Most mobile operating systems and third-party virtual keyboards use similar text editing features. Here, we focus mostly on the Google Android OS and its default keyboard, Gboard [30], behaviors.

Text Selection. Android enables users to directly position the cursor by touching the screen. The cursor is placed where the finger lands. Sliding the finger in any direction moves the cursor along with the finger within the text. A magnifier window appears to display the text under the finger to

(a) Magnifier window　　　(b) Text selection handles and the edit menu　　　(c) Gboard's text editing layout
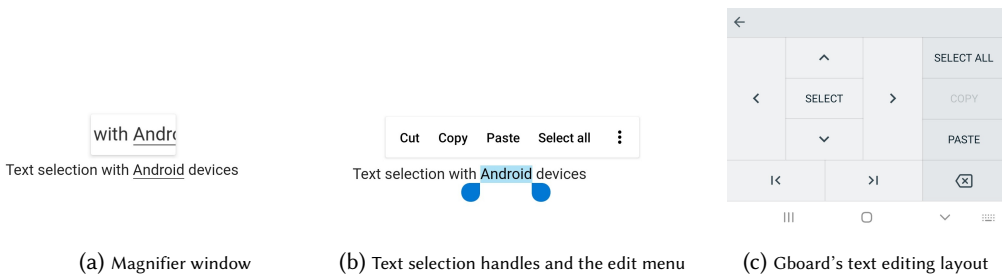
Fig. 2. Some cursor positioning, text selection, and editing features on Android-based devices.

facilitate precise target selection (Fig. 2a). Newer Android devices enable users to place the finger on the Space and slide it left or right to move the cursor horizontally within the text [16]. Some Apple devices enable using the whole keyboard as a trackpad by long-tapping or applying extra force on the Space [32]. Long-tapping on text for 500–1,000 ms automatically selects the word under the finger and displays two text selection handles [15] (Fig. 2b). Users then can adjust the handles to modify text selection. Gboard also includes a dedicated text editing keyboard layout to enable users to move the cursor and select text with virtual keys (Fig. 2c).

Text Editing. Text editing on mobile devices is straightforward. Upon selection of text, a menu containing the options cut, copy, and paste appears above the text (Fig. 2b). Users then select an option to perform the corresponding task. Users could also use the dedicated keyboard layout for editing (see Fig. 2c) to perform these tasks. To reposition (or move) the text, users long-tap on selected text, then move the finger to the desired location.
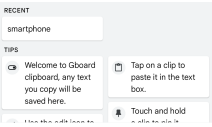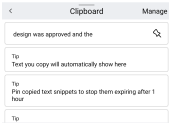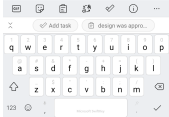
Text Formatting. Most mobile operating systems do not provide dedicated methods for text formatting, instead rely on the developers to include text formatting features in applications. Some applications display a text formatting tool above the virtual keyboard to bold, italicize, and underline the selected text (Fig. 3a). Web developers could also force Android to display some formatting options in the edit menu (Fig. 3b), accessed by tapping on the kebab menu icon (⋮).

Default Clipboard. Gboard has recently included a clipboard feature to enable multiple cuts, copies, and pastes [37]. Once this feature is enabled from the settings, it displays the last cut or copied text in the suggestion bar. It also stores the last five cut and copied text for users to paste (Fig. 3c). This feature is particularly useful when users want to first cut or copy multiple chunks, then paste them when done (rather than repeated "cut/copy-paste" sequences). To access the recorded chunks, users expand the keyboard menu by pressing an arrowhead icon (<), then select the clipboard option. We reviewed the most popular virtual keyboards for smartphones, including Google Gboard, Apple iOS keyboard, Microsoft Swiftkey, Samsung keyboard, Fleksy, Grammarly, and Typewise, by using these on our devices for several days. It revealed that this feature is not supported by all keyboards. Table 2 presents an excerpt of the findings.

4 GESHORT TEXT EDITING AND FORMATTING BEHAVIORS

We collected data from a prior study [9], where participants were asked to transcribe text using a virtual keyboard. We also conducted a new pilot study with 12 participants (3 female, 9 male),

Table 2. Availability of advanced clipboard features in popular virtual keyboards. "NA" signifies not available.

| Function | Gboard | | | Microsoft | Default |
	Android	Apple iOS	Samsung	SwiftKey	Apple iOS
Clipboard		NA	NA		NA
Clipboard Buffer		NA	NA		NA

(a) Text formatting on Outlook for Android   (b) Android's text formatting options   (c) Gboard's clipboard with all recorded text
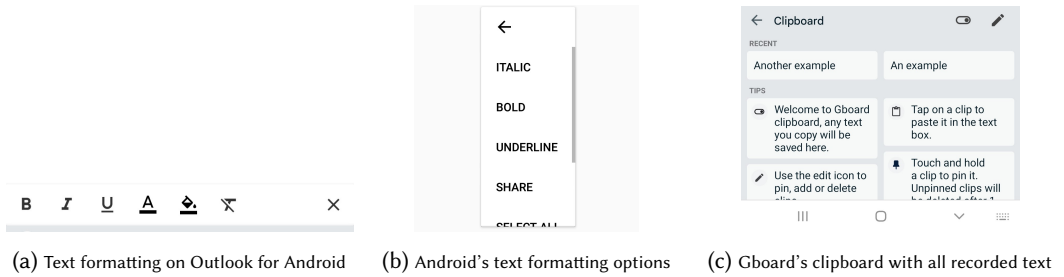
Fig. 3. Text formatting features in third-party applications and on Android-based devices, and the new clipboard feature of Gboard.

M = 27.92 years (SD = 4.87), where participants selected random chunks of text on a smartphone. After the pilot study, participants discussed their mobile text editing and formatting strategies and challenges. Based on the cursor positioning patterns identified in the studies, GeShort uses simple cursor positioning rules, and gesture-based cursor movement, text selection, editing, and formatting options, as well as a floating clipboard to enable faster and more accurate text editing and formatting on mobile devices.

4.1 GeShort Text Selection

We developed three simple rules to facilitate precise target selection.

(1) This rule is based on the observation that users usually cut or copy from the beginning of one word to the end of the same or a different word. When the finger lands within three characters from the beginning or the end of a word (see Section 4.2), the cursor is placed at the beginning or the end of the respective word. When both the beginning of one word and the end of another word are within the three-character threshold:

    1.1 The cursor is placed at the beginning of the respective word if it is the first cursor placement for text selection.

    1.2 The cursor is placed at the end of the respective word if it is the last cursor placement for text selection.

(2) This rule is based on the observation that for words with prefixes or suffixes, users tend to cut, copy, or delete the prefix or suffix. If the finger lands closer to the middle of a word that is either a gerund (words with "-ing"), compound (blends of multiple words, like "guessability" is a blend of "guess" and "ability"), plural form of a word (like "boxes"), or contraction (like "couldn't"), the cursor is placed between the prefix and the suffix. For example, if the finger lands closer to the middle of the word "guessability", the cursor is placed between "guess" (prefix) and "ability" (suffix). Our experimental prototype uses a dictionary to identify these words.

(3) In all other cases, the method behaves like the default cursor positing method, described in Section 3.

GeShort enables users to move the cursor by one character at a time by swiping left and right (horizontal movement), and one line at a time by swiping up and down (vertical movement) on the text area. For text selection, users first position the cursor at the beginning of the intended text and double-tap to initiate selection. Users then re-position the cursor at the end of the text and double-tap to confirm the selection (Fig. 1). With this approach, users have to precisely position the cursor to indicate the start and the end of a selection, but the double-taps to confirm the selection do not have to be precise, rather tapping closer to the cursor is sufficient (within 20 pixels). We

acknowledge that some mobile applications use double-taps to enable the selection of a whole word or a chunk of text, and to trigger the zoom-in feature on a document or website. We resolve the first conflict by enabling the selection of a whole word or chunk of text by long-tapping (500 ms) on the text. The second conflict can be avoided by tapping on the gutter or unused areas of a document or a website (i.e., not on text). To cancel or re-start a selection process, users double-tap on the cursor again. Algorithm 1 describes GeShort's text selection procedure.

Algorithm 1: GeShort Cursor Positioning

Input: Touch position t
Function CursorPlacement(t):
    $c_{default} \leftarrow$ default cursor position closest to the touch t
    if $c_{default}$ *points to whitespace* **then**
       | **return** $c_{default}$
    $w \leftarrow$ the word that contains cursor $c_{default}$
    $s_w, e_w \leftarrow$ beginning and end cursor coordinates of word w
    if $pixels(c_{default} - s_w) < 20$ **then**
       | **return** s_w
    else if $pixels(e_w - c_{default}) < 20$ **then**
       | **return** e_w
    else if w *is gerund* **then**
       | **return** $e_w - 3$
    else if w *is plural* **then**
       if w *ends with 's'* **then**
          | $e_w - 1$
       else if w *ends with 'es'* **then**
          | $e_w - 2$
    else if w *is portmanteau or contraction* **then**
       $s \leftarrow$ split point of the words w
       return s
    return $c_{default}$

4.2 Three-Character Threshold

The three character threshold in Rule (1) is picked based on the mean touch contact area of the thumb (w: $6 \times h$: 7.2 mm) and the index finger (w: $5.5 \times h$: 6.7 mm) [44], which occludes about three characters in the default font type and size on most smartphones. Android OS, for instance, use the font Roboto at 18 sp, which takes on average w: $2.3 \times h$: 3.0 mm screen per character.

Since this threshold and the suffix-prefix cursor positioning in Rule (2) seldom forces users to swipe left or right to move the cursor to the right position (when an incorrect position is selected), we conducted a keystroke-level analysis to find out whether correction efforts outweigh their benefits. Correcting cursor position with GeShort can be described in the following keystroke-level model $t_P + (n \times t_K)$, where t_P is the time (ms) to position the cursor with two taps, t_K is the time to perform a swipe gesture, and n represents character offset, which is the total number of characters between the current and the target position. The model does not account for homing time (t_H, the time to move the finger to the target) and the mental preparation time (t_M, the time to visually scan the display, and prepare for the next task) for simplicity as these parameters are difficult to gather and separate from other parameters in user studies. Since these parameters are likely to be comparable between the methods, the model is still capable to estimate performance differences (%) between them.

We conducted a pilot study to collect the parameter values. Six participants (M = 26.7 years, SD = 6.1) took part in the pilot study. Four of them identified as female and two as male. They were all experienced smartphone users (M = 8.33 years of experience, SD = 2.3). In the study, they held a smartphone with their dominant hand, then performed tap and the two swipe gestures using the thumb of the same hand. In order to collect direct cursor positioning time with the default method, they also precisely positioned the cursor at randomly selected positions in a paragraph using the same holding posture. Each task were performed 12 time, with ~5 seconds breaks in between (6 × 12 × 4 = 288 data points). The study identified the following parameter values: $t_P = 320$, $t_K = 600$. Precise cursor positioning with the default method took on average 2,166 ms (SE = 356) including correction efforts (53% of all attempts required repositioning the cursor to the correct position). We then predicted cursor positioning and adjustment time with GeShort with $n = 0$ to 3. We did not consider an offset over three characters ($n > 3$) since the average word length in the English language is about 5 characters [13, 33], which, combined with Rules (1) and (2), assures that the offset will almost never be over three characters. Table 3 presents the predicted positioning and adjustment time with GeShort, where one can see that cursor re-positioning with three character offset is still 2% lower than the average 2,166 ms cursor positioning time with the default method. This further motivates the work.

Table 3. Predicted cursor positioning and adjustment time with GeShort, together with estimated performance gain in relevance to the average cursor positioning time with the default method.

Character Offset	GeShort $t_P + (n \times t_K)$	Performance Gain
$n = 0$	320 ms	85%
$n = 1$	920 ms	58%
$n = 2$	1,520 ms	30%
$n = 3$	2,120 ms	2%

4.3 GeShort Text Editing and Formatting

GeShort uses gestural shortcuts, designed based on keyboard hotkeys, for text editing and formatting. Table 4 presents these shortcuts and their desktop counterparts. These shortcuts replace the "Ctrl" or "Cmd" of keyboard shortcuts with a gesture initiated from the Space. For example, to copy a selected chunk of text, users swipe from the Space to the "C" key. Since these gestural shortcuts are initiated from the Space (Figs. 1, 5), they do not interfere with gesture-based text entry techniques like ShapeWriter or Swipe [35, 46]. Users can undo the last performed task by performing the undo shortcut (Space→Z).

Table 4. Proposed gestural shortcuts for text editing and formatting and their desktop counterparts. A '→' signifies a gesture.

Shortcuts	Cut	Copy	Paste	Bold	Italic	Underline	Undo	Redo
Keyboard	Ctrl + X	Ctrl + C	Ctrl + V	Ctrl + B	Ctrl + I	Ctrl + U	Ctrl + Z	Ctrl + Y
Gestures	Space→X	Space→C	Space→V	Space→B	Space→I	Space→U	Space→Z	Space→Y

(a) Bold   (b) Italicize   (c) Underline
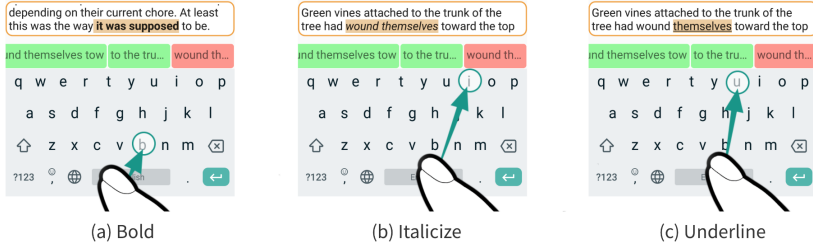
Fig. 4. Gestural shortcuts to (a) bold, (b) italicize, and (c) underline text, respectively. The shortcuts are initiated from the Space, thus do not interfere with gesture-based text entry techniques like ShapeWriter or Swipe.

4.4   GeShort Move Function

GeShort enables users to move text to a desired position, a feature not supported by most alternative approaches. Upon selection of a chunk of text, users can convert the virtual keyboard to a trackpad by holding the finger on the Space. When enabled, the keyboard is greyed out with all keys disabled. Users then move the finger over the trackpad to reposition the cursor to the intended position, and lift the finger to move the selected text to the cursor position. The design of this function is inspired by the drag and drop feature on desktop platforms, which enables users to "grab" an object to "drag" it to a different location. This is because, research [12] showed that exploiting specific knowledge that users already have of other domains (i.e., (re)using metaphors) facilitates learning.

(a)   (b)   (c)   (d)
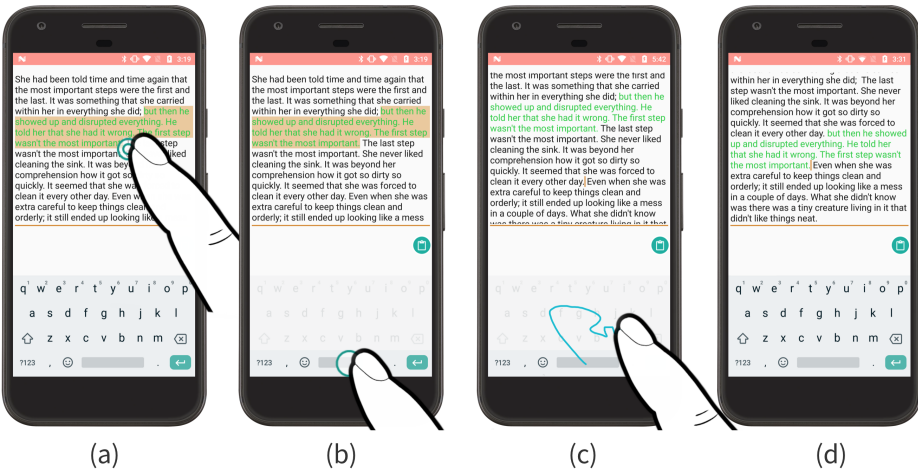
Fig. 5. Moving text with GeShort: the user (a) selects text, (b) long-taps on the Space to turn the keyboard into a trackpad, (c) moves the finger over the trackpad to reposition the cursor, (d) lifts the finger to place the selected text to the cursor position.

4.5   GeShort Floating Clipboard

GeShort provides easy access to the clipboard to facilitate delayed, repeated, and batch editing tasks. Unlike the default clipboard, which users have to access by extending the keyboard options,
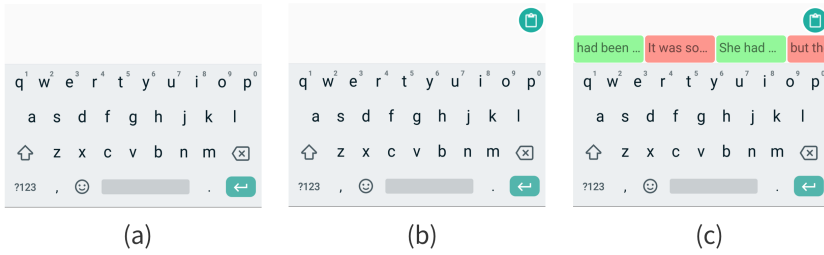
Fig. 6. Different states of GeShort's floating clipboard: (a) the clipboard icon is not visible when it is empty, (b) the clipboard icon appears when it has content, (c) tapping on the icon displays recently cut and copied text in a bar above the keyboard, highlighted in red and green backgrounds, respectively.

GeShort automatically displays a clipboard icon when it has cut or copied text (Fig. 6). Users tap on the icon to see snippets (i.e., the first nine characters of the text) of the content in a bar, then tap on a snippet to paste the corresponding text. Once opened, the clipboard bar remains visible until users tap on the icon again to hide it. The bar distinguishes cut and copied text using red and green backgrounds, respectively. This is based on prior research that showed that distinguishing different types of text editing actions using different colors improves usability and performance [2, 10]. Users can swipe left and right on the bar to access cut and copied text that are not visible in the bar. Users can extend a snippet by long-tapping on it. The text shrinks back to preview size upon lifting the finger to accommodate more excerpts in the bar. GeShort supports the following gestural shortcuts to afford users better control of its behavior, which are not supported by the default clipboard: Space→"V" pastes the last cut or copied text, Space→Enter pastes all items from the clipboard (to enable batch pasting with one action), and Space→Delete clears the clipboard.

5 USER STUDY 1: DEFAULT VS. GESHORT

We conducted a user study to compare the basic editing (cut, copy, paste, move, and delete) and formatting (bold, italic, underline) tasks with the default Android keyboard Gboard and GeShort.

5.1 Apparatus

We used a Motorola Moto G^5 Plus smartphone ($150.2 \times 74 \times 7.7$ mm, 155 g) at 1080×1920 pixels running on Android OS 7.0 in the user study. We developed a custom application with Android Studio 3.5.1, SDK 24 to display the tasks and record all interactions with timestamps. The floating clipboard was disabled in this study since only basic editing tasks were investigated. The application had four parts: a paragraph at the top, followed by a section presenting one task at a time, a text area to paste the cut/copied text, and a virtual keyboard (Fig. 7).

5.2 Participants

Twelve participants aged from 20 to 32 years (M = 24.83, SD = 4.13) took part in the study. They were recruited through social networking platforms, e-mailing lists, and word of mouth. Eight of them identified themselves as female and four as male. All of them were experienced mobile users (M = 8.04 years of experience, SD = 3.84). Ten of them were owners of Apple iOS-based smartphones, on which they used either the default Apple keyboard (N = 8) or Gboard (N = 2). The remaining two were Android OS-based smartphone owners, on which they used either the default Samsung keyboard (N = 1) or Gboard (N = 1). They all were proficient in the English language (native, bilingual, or advanced-level, moderate speakers of the language). All of them were right-handed.
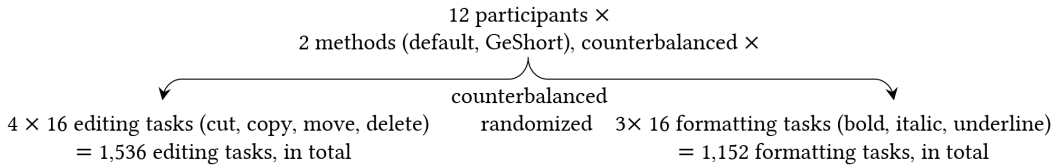
Fig. 7. The device with GeShort (left) and the default method (right), and two volunteers participating in the study.

They all received U.S. $15 for participating in the study. Fig. 7 illustrates the custom applications and two volunteers participating in the study.

5.3 Design

The study used a within-subjects design with two independent variables: 1) *method* with two levels: default Gboard and GeShort and 2) *task* with two levels: editing and formatting. The design was as follows:

<div align="center">

12 participants ×
2 methods (default, GeShort), counterbalanced ×

counterbalanced
randomized

4 × 16 editing tasks (cut, copy, move, delete)   3× 16 formatting tasks (bold, italic, underline)
= 1,536 editing tasks, in total   = 1,152 formatting tasks, in total

</div>

5.3.1 Task Selection. We carefully selected the experimental tasks to increase the external validity of the study. In the editing phase, participants performed sixteen cut, copy, move, and delete tasks with each method in randomized order. While in the formatting phase, they performed sixteen bold, italic, and underline tasks with each method in randomized order. The text to be edited or formatted were equally split between word-level and phrase-level selection tasks, the former composed of 1–3 words and the latter composed of multiple lines of text. Word-level and phrase-level selection tasks were further divided into mid-selection and start/end-selection tasks, the former required selecting text within a word and the latter required selecting text from/to the start/end of a word. For the tasks, we generated eight random paragraphs using a freely available service[1]. We used random paragraphs to reduce the effects of the content and the context of the paragraphs on the selection tasks. Using excepts from existing sources could have introduced a confound since participants' familiarity to the text cannot be predicated ahead of the study. Table 5 presents four examples of these tasks. Table 6 presents statistics about the paragraphs.

5.3.2 Performance Metrics. The dependent variables were the following performance metrics.

- **Task completion time (s)** is the average time (in seconds) participants took to complete one task. For deeper analysis, we divided task completion time into selection time and action time, representing the average time to select text and the average time to perform an editing or formatting task, respectively. Hence, *task completion time = selection time + action time.*

---

[1]Random Paragraph Generator: https://randomwordgenerator.com/paragraph.php.

Table 5. Examples of experimental tasks of different selection patterns.

Selection Level	Task	Example
Word-level start/end-selection	Copy	**Copy and Paste words beginning** She had been told time and time again that the most important steps were the first and
Word-level mid-selection	Cut	**Cut and Paste words middle** was captivating by reason of a certain frankness of expression and a contradictory
Phrase-level start/end-selection	Move	**Move sentences beginning** could see movement. She squinted her eyes and peered in the direction of the movement; trying to decipher exactly what she had spied.
Phrase-level mid-selection	Bold	**Bold sentences middle** happened. It didn't even really make sense to him. All he knew was that he froze at the moment; and nothing in his body allowed to move. It was as if he had instantly become

- **Errors per task** represents the average character errors committed per selection task. This metric is comparable to the minimum string distance measure in text entry research that represents the similarity between two text sequences using the Levenshtein distance algorithm [31]. The distance is defined as the minimum number of primitive operations (namely insertion, deletion, and substitution) needed to transform a cut or copied text to the text presented in the task [40].

5.4 Procedure

The study was conducted in a quiet room, one participant at a time. Upon arrival, we explained the research and collected their informed consent forms and demographics. Participants sat in front of a desk. They were asked to hold the device with the dominant hand in a comfortable position and interact with the thumb of the same hand. We then demonstrated the method used in the first condition (the conditions were counterbalanced) and asked them to practice all features by performing at least two tasks per feature. They could extend the duration of the practice by one additional task per feature upon request. We then started the first condition, where they were asked to perform the tasks as fast and accurate as possible. The custom application displayed one task at a time. In addition to written instructions, the relevant parts of the paragraphs were highlighted

Table 6. Statistics about the randomly generated paragraphs used in the studies.

Paragraph ID	Total Character	Total Word	Total Line
1	419	78	10
2	445	84	11
3	498	92	12
4	519	88	13
5	398	80	10
6	425	83	11
7	464	89	13
8	496	90	12
Mean	458	86	12

in different colored font to reduce the visual scan time: green for copy and move, red for cut and delete. Formatting tasks were highlighted in a purple-colored font (Table 5). After completing a task, participants tapped on the Next key to see the next task. Correcting selection errors was not required. After they completed all tasks in the condition, we demonstrated the next method, asked them to practice with it, then started the next condition. We asked participants to practice with both methods, although they were familiar with Gboard, to make sure they know how to use the default editing features of the keyboard. In the practice, participants used all features by performing at least two tasks per feature (see Section 5.3.1). Participants were instructed to read the presented task carefully before initiating a task. The tasks used in the training were not repeated in the main study.

During the study, we did not restrict participants from using any default features, including the clipboard or cursor movements by swiping on the space key. Yet, participants almost never used these features. We enforced a 2-minute break after each condition to avoid the effect of fatigue. Participants could extend the duration of the breaks by 2 extra minutes upon request. After the study, participants completed the NASA-TLX questionnaire to rate the methods' perceived workload [26] and a custom questionnaire to rate the perceived speed, accuracy, ease-of-use, integration of the functions, and willingness-to-use on a 5-point Likert scale. For analysis, we calculated the raw TLX scores by individual sub-scales, which is a common practice in the literature [25].

All researchers involved in this study were fully vaccinated for COVID-19. All participants were pre-screened for COVID-19 symptoms during the recruitment process by a researcher, and on the day of the study by the host institute. Both the researcher and the participants wore face coverings and sanitized their hands before a session. The researcher also maintained a 3-feet distance from the participants at all times. All devices and surfaces were disinfected before and after each session. This protocol was reviewed and approved by the UC Merced Institutional Review Board (IRB).

5.5 Results

A complete study session took about 60 minutes, including demonstration, questionnaires, and breaks. A Shapiro-Wilk test revealed that the response variable residuals were normally distributed. A Mauchly's test indicated that the variances of populations were equal. We did not exclude any outliers from the analysis. We used a repeated-measures ANOVA for the quantitative within-subjects factors and a Wilcoxon Signed-Rank test for the questionnaire data.

Fig. 8. The average overall, editing, and formatting task completion time with the two methods segmented in selection and action time. Error bars represent ±1 standard error (SE).

5.5.1 Task Completion Time. An ANOVA identified a significant effect of method on task completion time ($F_{1,11} = 17.29$, $p < .005$). On average, the default and GeShort took 10.56 s (SE = 0.20) and 8.99 s (SE = 0.20) to complete a task, respectively. The effects on selection time ($F_{1,11} = 11.00$, $p < .01$)

and action time ($F_{1,11} = 13.87, p < .005$) were also statistically significant. The 5.7 s and 6.9 s selection time values are higher than the predicted values in Table 3 as the latter did not account for the homing or the mental preparation time. The 17% performance gain identified here falls between the gains predicted for 2–3 character offsets. We further analyzed the data after filtering for editing and formatting tasks, where an ANOVA identified significant effects of method on both editing ($F_{1,11} = 5.44, p < .05$) and formatting ($F_{1,11} = 76.97, p < .0001$) task completion time. Fig. 8 illustrates the average overall, editing, and formatting task completion time split into selection and action time.
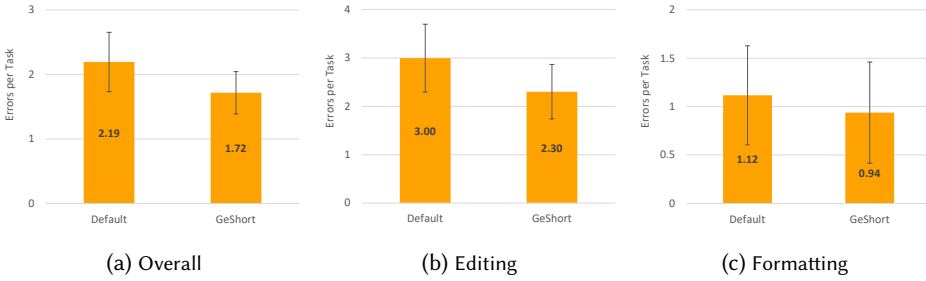

(a) Overall   (b) Editing   (c) Formatting

Fig. 9. The average overall, editing, and formatting errors per task with the two methods. Error bars represent ±1 standard error (SE).

5.5.2   Errors per Task. An ANOVA failed to identify a significant effect of method on errors per task ($F_{1,11} = 0.39, p = .54$). On average the default and GeShort caused 2.19 (SE = 0.46) and 1.72 (SE = 0.33) errors per task, respectively. We also failed to identify significant effects when filtered the data for editing ($F_{1,11} = 0.44, p = .52$) and formatting ($F_{1,11} = 0.04, p = .84$) tasks. Fig. 9 illustrates the average overall, editing, and formatting errors per task.

5.5.3   Perceived Workload. A Wilcoxon Signed-Rank test identified a significant effect of method on physical demand ($z = −2.14, p < .05$), effort ($z = −2.52, p <= .05$), and frustration ($z = −2.95, p < .005$). However, no significant effect was identified on mental demand ($z = −0.28, p = .78$), temporal demand ($z = −0.53, p = .69$), or performance ($z = −0.71, p = .48$). Fig. 10a illustrates median raw NASA-TLX ratings of both methods.

5.5.4   Usability. A Wilcoxon Signed-Rank test identified a significant effect of method on perceived speed ($z = −2.71, p < .01$), ease-of-use ($z = −2.39, p < .05$), function ($z = −2.38, p < .05$), and willingness-to-use ($z = −2.7, p < .01$). However, no significant effect was identified on perceived accuracy ($z = −1.85, p = .06$). Fig. 10b illustrates median user ratings of both methods.

5.6   Discussion

Participants were significantly faster with GeShort than Gboard (15% faster). This performance gain cannot be directly compared with the performance reported in the literature (Table 1) since those works were evaluated with much simpler tasks (did not consider all selection scenarios described in Section 5.3.1), compared only a subset of edit options (did not compare cut or move), or did not support one-handed interactions. An analysis revealed that participants performed both text selection and editing/formatting tasks significantly faster than the default method (11% and 17% faster, respectively), which suggests that both the proposed selection and editing/formatting approaches contributed to the faster task completion time of GeShort. When filtered the data for editing and formatting tasks, we found out that both types of tasks were performed significantly
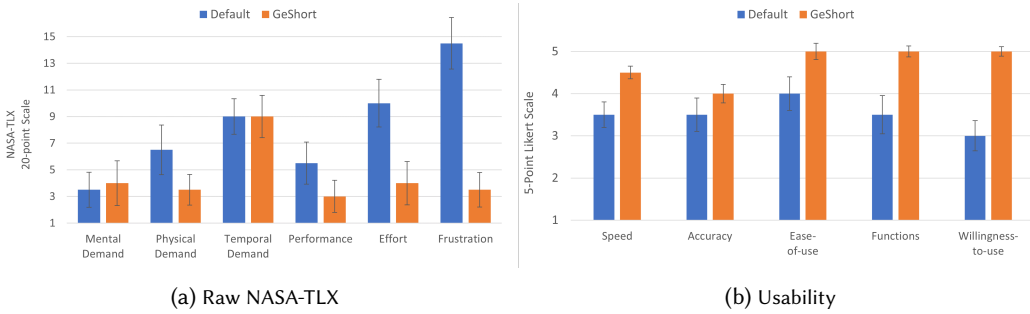
(a) Raw NASA-TLX

(b) Usability

Fig. 10. Median user ratings of both methods on perceived workload and usability questionnaires. Error bars represent ±1 standard error (SE).

faster with GeShort than the default method (11% faster editing and 22% faster formatting). Error rates of the two methods were not statistically significant.

The results of subjective feedback are also encouraging. In the perceived workload questionnaire (Fig. 10a), participants found GeShort to be significantly less physically demanding, which suggests that it better facilitates text editing and formatting with one hand than the default method. One participant (female, 20 years) commented that the default method *"was more physically and mentally challenging"* because its selection and editing/formatting features were not suited for one-handed interaction. Another participant (female, 31 years) praised GeShort saying that it *"was more precise in terms of text selection."* They also felt that the default method required significantly more effort, causing significantly more frustration performing the tasks. One participant (female, 26 years) commented, *"frustration occurred mostly when selecting with Google method."* Participants found both methods somewhat comparable in terms of metal and temporal demands.

In the usability questionnaire (Fig. 10b), participants found GeShort to be significantly faster and easier to use than the default method. One participant (female, 20 years) commented, *"The underline, bold, italic in GeShort was awesome! As someone who edits papers and writes essays on my phone I would actually use underline and such more often."* They felt that various functions were much better integrated in GeShort than the default method, making it substantially easier to use than the baseline. One participant (female, 26 years) commented that *"[various editing] functions [are] a lot easier to use with GeShort."* As a result, participants preferred using GeShort on their mobile devices significantly more than the default method.

6   USER STUDY 2: DEFAULT VS. GESHORT CLIPBOARD

We conducted a second user study to compare the advanced clipboard features (delayed and repeated pasting from the clipboard) of the default Android keyboard Gboard and GeShort.

6.1   Apparatus & Participants

The study used the same apparatus as the first study (Section 5.1). Twelve new participants took part in this study. They were recruited through social networking platforms, e-mailing lists, and word of mouth. Their age ranged from 18 to 31 years (M = 22.30, SD = 3.42). Six of them identified themselves as female and six as male. All of them were experienced mobile users (M = 9.46 years of experience, SD = 4.26). They all were owners of Apple iOS-based smartphones, on which they used the default Apple keyboard. All of them were proficient in the English language (native, bilingual, or advanced-level speakers of the language). Eleven of them were right-handed mobile users, while

one was left-handed. They all received U.S. \$15 for participating in the study. Fig. 11 illustrates the custom application and two volunteers participating in the study.
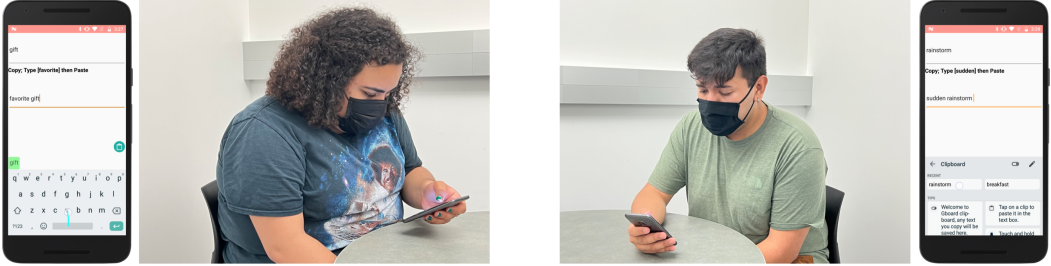

Fig. 11. The experimental device with GeShort and the floating clipboard (left) and the default clipboard (right), and two volunteers using the respective methods in the second user study.

6.2 Design & Procedure

The study used a within-subjects design with one independent variable *method* with two levels: default Gboard and GeShort. The design was as follows:

> 12 participants ×
> 2 methods (default, GeShort), counterbalanced × 20 tasks
> = 480 clipboard tasks in total.

6.2.1 Task Selection. Like the first user study, we carefully selected the experimental tasks to increase the external validity of the study. In the study, participants performed twenty clipboard tasks with each method in randomized order. Four different types of tasks were selected: 1) cut/copy, then paste from the suggestion bar, 2) cut/copy, type a word, then paste from the clipboard, 3) paste existing entries from the clipboard, 4) paste the same text 2–5 times (repeated paste). These tasks were selected to replicate real-life scenarios. For example, assume a user copied an address to share with a friend via text message. She could either paste it immediately (task 1), paste it after typing a message like *"here's the address"* (task 2), share the address later from the clipboard (task 3), or share it with multiple friends (task 4). To avoid potential confounding factors, the experimental tasks involved only one word that users could select by long-tapping on it (to eliminate the need for precise target selection), and pre-populated the clipboard with five words (to reduce the effect of visual scan time). Table 7 presents examples of these tasks.

6.2.2 Performance Metrics. The dependent variables were the task completion time and the errors per task performance metrics described in Section 5.3.2. However, unlike the first study, we did not split the task completion time by actions as performing these tasks required performing different numbers and combinations of actions.

6.2.3 Procedure. The study used the same procedure as the previous study (Section 5.4).

6.3 Results

A complete study session took about 30 minutes, including demonstration, questionnaires, and breaks. A Shapiro-Wilk test revealed that the response variable residuals were normally distributed. A Mauchly's test indicated that the variances of populations were equal. We did not exclude any outliers from the analysis. We used a repeated-measures ANOVA for the quantitative within-subjects factors and a Wilcoxon Signed-Rank test for the questionnaire data.
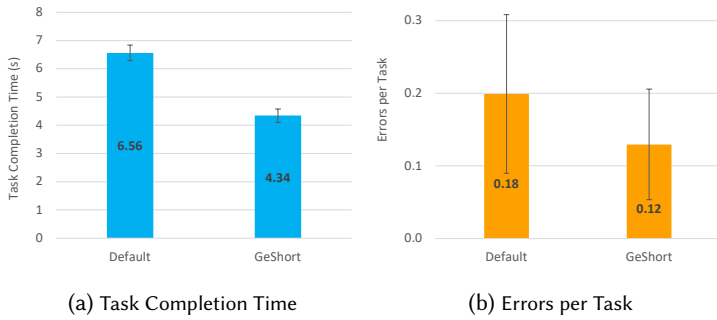
(a) Task Completion Time

(b) Errors per Task

Fig. 12. Average task completion time (s) and errors per task with two methods. Error bars represent ±1 standard error (SE).

6.3.1 Task Completion Time. An ANOVA identified a significant effect of method on task completion time ($F_{1,11} = 54.03, p < .0001$). On average the default and GeShort took 6.56 s (SE = 0.28) and 4.34 s (SE = 0.23) to complete a task, respectively (Fig. 12a). A Tukey-Kramer multiple-comparison test revealed that task completion time for editing tasks 2, 3, and 4 were significantly faster with GeShort than the default method, while task 1 was somewhat comparable. Fig. 13 illustrates this behavior. Section 6.2.1 describes the tasks.

6.3.2 Errors per Task. An ANOVA failed to identify a significant effect of method on errors per task ($F_{1,11} = 0.57, p = .46$). The default and GeShort caused 0.18 (SE = 0.10) and 0.12 (SE = 0.07) errors per task, respectively (Fig. 12b).

Table 7. Examples of the four types of clipboard tasks used in the second study.

Task	GeShort	Gboard
Cut/Copy, then paste from suggestion bar	**Cut/Copy and Paste from prediction bar** regular	**Cut/Copy and Paste from prediction bar** breakfast
Cut/Copy, type, then paste from suggestion bar	**Cut/Copy; Type [favorite] then Paste** favorite gift	**Cut/Copy; Type [took] then Paste** took quite
Paste existing entries from the clipboard	**Paste from clipboard** forces	**Paste from clipboard** together
Paste text multiple times (repeated paste)	**Repeated paste 2** believe believe	**Repeated paste 3** truth truth truth

6.3.3   Clipboard-specific Analysis. In the study, only the treatment group was allowed to use the gestural shortcuts, while the control group used the default selection methods. Hence, the above results present the combined benefits of the clipboard and the gestural shortcuts. We conducted a deeper analysis to compare only the performance of the two clipboard methods. For this, we filtered the data for the tasks that do not require using the shortcuts for text selection and editing, namely tasks 3 and 4 (50% of the total data). An ANOVA identified a significant effect of clipboard method on task completion time ($F_{1,11} = 54.65, p < .0001$). On average the default and the floating clipboard took 4.64 s (SE = 0.2) and 2.00 s (SE = 0.13) to complete a task, respectively. An ANOVA failed to identify a significant effect of clipboard method on errors per task ($F_{1,11} = 0.00, p = .95$). Both clipboard methods caused about 0.2 errors per task.
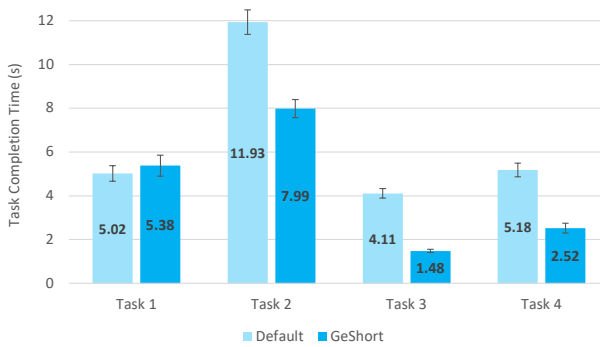

Fig. 13.   Average task completion time (s) per clipboard task. Error bars represent ±1 standard error (SE)

6.3.4   Perceived Workload. A Wilcoxon Signed-Rank test identified a significant effect of method on mental demand ($z = -2.20, p < .05$), physical demand ($z = -2.81, p < .01$), performance ($z = -2.67, p < .01$), effort ($z = -2.40, p <= .05$), and frustration ($z = -2.61, p < .01$). However, no significant effect was identified on temporal demand ($z = -0.93, p = .35$). Fig. 14a illustrates median raw NASA-TLX ratings of both methods.

6.3.5   Usability. A Wilcoxon Signed-Rank test identified a significant effect of method on perceived speed ($z = -3.13, p < .005$), ease-of-use ($z = -2.99, p < .005$), function ($z = -2.74, p < .01$), and willingness-to-use ($z = -2.65, p < .01$). However, no significant effect was identified on perceived accuracy ($z = -1.93, p = .05$). Fig. 14b illustrates median user ratings of both methods.

6.4   Discussion

Participants were significantly faster with GeShort than the default method (34% faster). They were also significantly faster when using only the clipboard features (35% faster). A deeper investigation revealed that participants were significantly faster in performing in-text pasting (task 2), delayed pasting (task 3), and repeated pasting (task 4) with GeShort than the default method (33%, 64%, and 51% faster, respectively), while performance of immediate pasting tasks (task 1) was relatively comparable between the methods (Fig. 13). Error rates of the two methods were not statistically significant. These results suggest that GeShort better facilitates performing advanced editing tasks than the most current features of the state-of-the-art.

   Results of subjective feedback also support this. In the perceived workload questionnaire (Fig. 14a), participants found GeShort to be significantly less demanding both mentally and physically. They
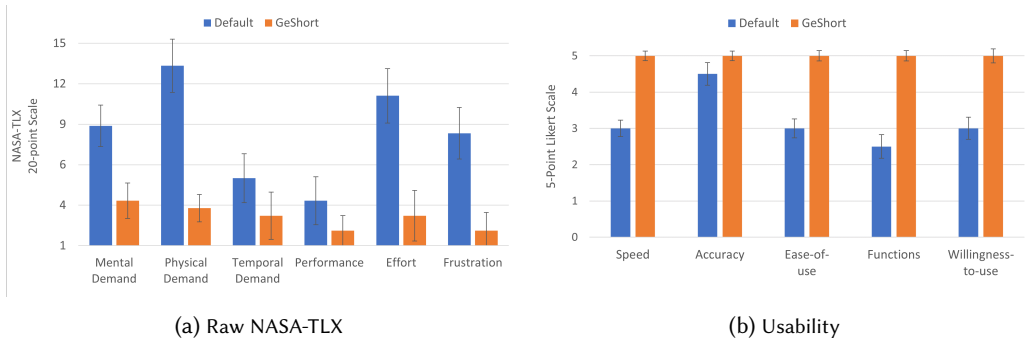
(a) Raw NASA-TLX

(b) Usability

Fig. 14. Median user ratings of both methods on perceived workload and usability questionnaires. Error bars represent ±1 standard error (SE).

also found the default method significantly more demanding than GeShort in terms of effort. They felt that the reduced mental and physical demands and effort significantly improved their performance with GeShort. One participant (female, 22 years) summarized, *"It took longer [with the default method] than anticipated! I never realized how many steps I had to take to do copy and paste. Using the GeShort method was a lot of simpler than imagined. The more I used it, the easier it gets."*

In the usability questionnaire (Fig. 14b), participants found GeShort to be significantly faster and easier to use than the default method. They felt that various functions were better integrated in GeShort than the default method, making it substantially easier to use than the baseline. Particularly, they appreciated how the gestural shortcuts matched the keyboard shortcuts. One participant (male, 21 years) commented that this made performing editing tasks *"much easier on GeShort."* Hence, participants preferred using GeShort on their mobile devices significantly more than the default method. One participant (female, 23 years) summarized, *"I found [the] new method easier to use and noticed I typed faster. GeShort made typing more accessible in my experiences especially when compared to Google."*

7 CONCLUSION

We presented GeShort, a text editing and formatting method for mobile devices. GeShort facilitates direct cursor positioning by using three simple rules, one-handed text editing and formatting with gestural shortcuts inspired by keyboard hotkeys, and delayed, repeated, and batch editing using a floating clipboard. We compared GeShort with the state-of-the-art Gboard in two user studies. Results of the first study revealed that the proposed method reduces text editing time by 11% and text formatting time by 22%. When the tasks are broken down into text selection and editing actions, GeShort reduces selection time by 11% and action time by 17%, validating the benefits of the method's selection approach and gestural shortcuts. Results of the second study revealed that the floating clipboard outperforms the clipboard feature of Gboard by 34% for advanced editing tasks. Besides, in both studies, participants found GeShort less onerous in terms of mental demand, physical demand, and effort, improving their overall performance with the method. They perceived GeShort as faster and easier to use than Gboard, felt that its functions were better integrated, and wanted to keep using it on their mobile devices.

7.1 Limitations

We acknowledge several limitations of the work. First, GeShort uses gestural shortcuts, which are designed based on keyboard hotkeys. Hence, those who are unfamiliar with the hotkeys may

require extra time to learn the method. Limited discoverability of gestural interactions have long been a discussion in the literature [45]. Further investigations, particularly in-the-wild studies, are needed to find out whether users are able to discover and use the method in the real-world. Second, we used common editing and formatting tasks in the user studies, thus it is unclear whether the findings are generalizable to more complex editing and formatting tasks. Although it is doubtful that users would choose to perform such complex tasks on mobile devices. We encourage the research community to explore this further. Finally, we used convenience sampling for recruiting participants, which resulted in mostly tech-savvy young adults in the user studies. Therefore, the results reported here may not be generalizable to a more diverse population.

(a) Bold font   (b) Colored font   (c) Bold font & labels   (d) Colored font & labels

Fig. 15. Four key highlighting approaches will be explored in the future to facilitate gestural shortcut discovery and learning.

8 FUTURE WORK

In the future, we will develop a more sophisticated approach to facilitate text selection on mobile devices. For this, a longitudinal study will be conducted to understand users' text selection behaviors, then a machine learning model will be developed to predict potential selection actions. We will also explore the possibility of customizing the method for other touchscreen-based devices, particularly smartwatches and interactive tabletops.

8.1 Key Highlighting

The current interface does not provide any visual cues to facilitate the discovery and learning of the gestural shortcuts. In the future, we will investigate whether highlighting all possible shortcut keys as the user touches or initiates a gesture from the Space accommodates discovery or improves learning and performance since such approaches have been shown to improve input performance in the literature [11, 22, 23]. Particularly, four different highlighting approaches will be explored: 1) bold font, 2) colored font, 3) bold font with labels, 4) colored font with labels, as illustrated in Fig. 15.

REFERENCES

[1] Maria Giovanna Albanese, Gennaro Costagliola, Mattia De Rosa, and Vittorio Fuccella. 2022. A Technique to Improve Text Editing on Smartphones. IEEE Computer Society, 1–3. https://doi.org/10.1109/VL/HCC53370.2022.9833120

[2] Ohoud Alharbi, Ahmed Sabbir Arif, Wolfgang Stuerzlinger, Mark D. Dunlop, and Andreas Komninos. 2019. WiseType: A Tablet Keyboard with Color-Coded Visualization and Various Editing Options for Error Correction. In *Proceedings of Graphics Interface 2019 (GI 2019)*. Canadian Information Processing Society, Toronto, ON, Canada, Article 4, 1–10. https://doi.org/10.20380/GI2019.04 event-place: Kingston, Ontario.

[3] Jessalyn Alvina, Carla F. Griggio, Xiaojun Bi, and Wendy E. Mackay. 2017. CommandBoard: Creating a General-Purpose Command Gesture Input Space for Soft Keyboard. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology (UIST '17)*. Association for Computing Machinery, New York, NY, USA, 17–28. https://doi.org/10.1145/3126594.3126639

[4] Toshiyuki Ando, Toshiya Isomoto, Buntarou Shizuki, and Shin Takahashi. 2018. Press & Tilt: One-Handed Text Selection and Command Execution on Smartphone. In *Proceedings of the 30th Australian Conference on Computer-Human Interaction (OzCHI '18)*. ACM, New York, NY, USA, 401–405. https://doi.org/10.1145/3292147.3292178 event-place: Melbourne, Australia.

[5] Toshiyuki Ando, Toshiya Isomoto, Buntarou Shizuki, and Shin Takahashi. 2019. One-Handed Rapid Text Selection and Command Execution Method for Smartphones. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems (CHI EA '19)*. ACM, New York, NY, USA, LBW0224:1–LBW0224:6. https://doi.org/10.1145/3290607.3312850 event-place: Glasgow, Scotland Uk.

[6] Ahmed Sabbir Arif. 2010. Metrics for Multi-Touch Input Technologies. *arXiv:2009.13219 [cs]* 2020-09-28 (2010). http://arxiv.org/abs/2009.13219 arXiv: 2009.13219.

[7] Ahmed Sabbir Arif. 2012. A Survey on Mobile Text Entry Handedness: How Do Users Input Text on Handheld Devices While Nomadic?. In *2012 4th International Conference on Intelligent Human Computer Interaction (IHCI)*. 1–6. https://doi.org/10.1109/IHCI.2012.6481818

[8] Ahmed Sabbir Arif, Sunjun Kim, Wolfgang Stuerzlinger, Geehyuk Lee, and Ali Mazalek. 2016. Evaluation of a Smart-Restorable Backspace Technique to Facilitate Text Entry Error Correction. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 5151–5162. https://doi.org/10.1145/2858036.2858407 event-place: San Jose, California, USA.

[9] Ahmed Sabbir Arif and Wolfgang Stuerzlinger. 2013. Evaluation of a New Error Prevention Technique for Mobile Touchscreen Text Entry. In *Proceedings of the 25th Australian Computer-Human Interaction Conference: Augmentation, Application, Innovation, Collaboration (OzCHI '13)*. Association for Computing Machinery, New York, NY, USA, 397–400. https://doi.org/10.1145/2541016.2541063

[10] Ahmed Sabbir Arif, Cristina Sylla, and Ali Mazalek. 2016. Learning New Words and Spelling with Autocorrections. In *Proceedings of the 2016 ACM International Conference on Interactive Surfaces and Spaces (ISS '16)*. ACM, New York, NY, USA, 409–414. https://doi.org/10.1145/2992154.2996790 event-place: Niagara Falls, Ontario, Canada.

[11] Tyler Baldwin and Joyce Chai. 2012. Towards Online Adaptation and Personalization of Key-Target Resizing for Mobile Devices. In *Proceedings of the 2012 ACM international conference on Intelligent User Interfaces (IUI '12)*. Association for Computing Machinery, New York, NY, USA, 11–20. https://doi.org/10.1145/2166966.2166969

[12] Alan F. Blackwell. 2006. The Reification of Metaphor as a Design Tool. *ACM Transactions on Computer-Human Interaction* 13, 4 (Dec. 2006), 490–530. https://doi.org/10.1145/1188816.1188820

[13] Vladimir V. Bochkarev, Anna V. Shevlyakova, and Valery D. Solovyev. 2012. Average Word Length Dynamics as Indicator of Cultural Changes in Society. https://doi.org/10.48550/arXiv.1208.6109 arXiv:1208.6109 [cs].

[14] Francesco Cafaro, Leilah Lyons, and Alissa N. Antle. 2018. Framed Guessability: Improving the Discoverability of Gestures and Body Movements for Full-Body Interaction. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3173574.3174167

[15] Chen Chen, Simon T. Perrault, Shengdong Zhao, and Wei Tsang Ooi. 2014. BezelCopy: An Efficient Cross-Application Copy-Paste Technique for Touchscreen Smartphones. In *Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces (AVI '14)*. Association for Computing Machinery, New York, NY, USA, 185–192. https://doi.org/10.1145/2598153.2598162

[16] Jason Cipriani. 2016. Google's Secret Keyboard Feature Lets You Easily Move the Cursor. https://www.cnet.com/tech/services-and-software/googles-secret-keyboard-feature-gives-you-precise-cursor-control/

[17] Rajkumar Darbar, Arnaud Prouzeau, Joan Odicio-Vilchez, Thibault Lainé, and Martin Hachet. 2021. Exploring Smartphone-Enabled Text Selection in AR-HMD. In *Proceedings of Graphics Interface 2021 (GI 2021)*. Canadian Information Processing Society, Toronto, ON, Canada, 117 – 126. https://doi.org/10.20380/GI2021.14 ISSN: 0713-5424 event-place: Virtual Event.

[18] Alexander Keith Eady and Audrey Girouard. 2015. Caret Manipulation using Deformable Input in Mobile Devices. In *Proceedings of the Ninth International Conference on Tangible, Embedded, and Embodied Interaction (TEI '15)*. Association for Computing Machinery, New York, NY, USA, 587–591. https://doi.org/10.1145/2677199.2687916

[19] Katherine Fennedy, Angad Srivastava, Sylvain Malacria, and Simon T Perrault. 2021. Towards a Unified and Efficient Command Selection Mechanism for Touch-Based Devices Using Soft Keyboard Hotkeys. *ACM Transactions on Computer-Human Interaction* (2021). https://hal.archives-ouvertes.fr/hal-03319260 Publisher: Association for Computing Machinery.

[20] Vittorio Fuccella, Poika Isokoski, and Benoit Martin. 2013. Gestures and Widgets: Performance in Text Editing on Multi-Touch Capable Mobile Devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. Association for Computing Machinery, New York, NY, USA, 2785–2794. https://doi.org/10.1145/2470654.2481385

[21] Vittorio Fuccella and Benoît Martin. 2017. TouchTap: A Gestural Technique to Edit Text on Multi-Touch Capable Mobile Devices. In *Proceedings of the 12th Biannual Conference on Italian SIGCHI Chapter (CHItaly '17)*. Association for Computing Machinery, New York, NY, USA, 1–6. https://doi.org/10.1145/3125571.3125579

[22] Jun Gong, Bryan Haggerty, and Peter Tarasewich. 2005. An Enhanced Multitap Text Entry Method with Predictive Next-Letter Highlighting. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems (CHI EA '05)*. Association for Computing Machinery, New York, NY, USA, 1399–1402. https://doi.org/10.1145/1056808.1056926

[23] Asela Gunawardana, Tim Paek, and Christopher Meek. 2010. Usability Guided Key-Target Resizing for Soft Keyboards. In *Proceedings of the 15th international conference on Intelligent user interfaces (IUI '10)*. Association for Computing Machinery, New York, NY, USA, 111–118. https://doi.org/10.1145/1719970.1719986

[24] Carl Gutwin, Carl Hofmeister, David Ledo, and Alix Goguey. 2020. Learning Multiple Mappings: an Evaluation of Interference, Transfer, and Retention with Chorded Shortcut Buttons. In *Proceedings of Graphics Interface 2020 (GI 2020)*. Canadian Human-Computer Communications Society / Société canadienne du dialogue humain-machine, Toronto, Ontario, Canada, 206 – 214. https://doi.org/10.20380/GI2020.21 event-place: University of Toronto.

[25] Sandra G. Hart. 2006. Nasa-Task Load Index (NASA-TLX); 20 Years Later. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 50, 9 (Oct. 2006), 904–908. https://doi.org/10.1177/154193120605000909 Publisher: SAGE Publications Inc.

[26] 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, Amsterdam, The Netherlands, 139–183. https://doi.org/10.1016/S0166-4115(08)62386-9

[27] Erik D. Kennedy. 2018. The Android/Material Design Font Size Guidelines. https://learnui.design/blog/android-material-design-font-size-guidelines.html

[28] Andreas Komninos, Mark Dunlop, Kyriakos Katsaris, and John Garofalakis. 2018. A Glimpse of Mobile Text Entry Errors and Corrective Behaviour in the Wild. In *Proceedings of the 20th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct (MobileHCI '18)*. Association for Computing Machinery, Barcelona, Spain, 221–228. https://doi.org/10.1145/3236112.3236143

[29] Huy Viet Le, Sven Mayer, Maximilian Weiß, Jonas Vogelsang, Henrike Weingärtner, and Niels Henze. 2020. Shortcut Gestures for Mobile Text Editing on Fully Touch Sensitive Smartphones. *ACM Transactions on Computer-Human Interaction* 27, 5 (Aug. 2020), 33:1–33:38. https://doi.org/10.1145/3396233

[30] Reena Lee. 2016. Gboard, Now Available for Android. https://blog.google/products/search/gboard-now-on-android/ Library Catalog: blog.google.

[31] Vladimir I Levenshtein and others. 1966. Binary Codes Capable of Correcting Deletions, Insertions, and Reversals. In *Soviet physics doklady*, Vol. 10. MAIK Nauka/Interperiodica, Soviet Union, 707–710. Issue: 8.

[32] Andy Moser. 2021. The Simple iPhone Hack You Need to Fix Your Texting Typos. https://mashable.com/article/how-to-use-iphone-keyboard-trackpad-texting Section: Tech.

[33] Peter Norvig. 2013. English Letter Frequency Counts: Mayzner Revisited or ETAOIN SRHLDCU. http://norvig.com/mayzner.html

[34] Laxmi Pandey, Azar Alizadeh, and Ahmed Sabbir Arif. 2020. Enabling Predictive Number Entry and Editing on Touchscreen-Based Mobile Devices. In *Proceedings of the 2020 Conference on Human Information Interaction and Retrieval (CHIIR '20)*. Association for Computing Machinery, New York, NY, USA, 13–22. https://doi.org/10.1145/3343413.3377957

[35] Philip Quinn and Shumin Zhai. 2018. Modeling Gesture-Typing Movements. *Human–Computer Interaction* 33, 3 (May 2018), 234–280. https://doi.org/10.1080/07370024.2016.1215922 Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/07370024.2016.1215922.

[36] Volker Roth and Thea Turner. 2009. Bezel Swipe: Conflict-Free Scrolling and Multiple Selection on Mobile Touch Screen Devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 1523–1526. https://doi.org/10.1145/1518701.1518933

[37] Judy Sanhz. 2020. How to Use Gboard Clipboard in Android. https://www.technipages.com/how-to-use-gboard-clipboard-in-android

[38] Jean-Baptiste Scheibel, Cyril Pierson, Benoît Martin, Nathan Godard, Vittorio Fuccella, and Poika Isokoski. 2013. Virtual Stick in Caret Positioning on Touch Screens. In *Proceedings of the 25th Conference on l'Interaction Homme-Machine (IHM '13)*. Association for Computing Machinery, New York, NY, USA, 107–114. https://doi.org/10.1145/2534903.2534918

[39] Robin Schweigert, Jan Leusmann, Simon Hagenmayer, Maximilian Weiß, Huy Viet Le, Sven Mayer, and Andreas Bulling. 2019. KnuckleTouch: Enabling Knuckle Gestures on Capacitive Touchscreens using Deep Learning. In *Proceedings of Mensch und Computer 2019 (MuC'19)*. Association for Computing Machinery, New York, NY, USA, 387–397. https://doi.org/10.1145/3340764.3340767

[40] R. William Soukoreff and I. Scott MacKenzie. 2003. Metrics for Text Entry Research: An Evaluation of MSD and KSPC, and a New Unified Error Metric. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '03)*. ACM, New York, NY, USA, 113–120. https://doi.org/10.1145/642611.642632 event-place: Ft. Lauderdale, Florida, USA.

[41] R. W. Soukoreff and I. S. MacKenzie. 2009. An Informatic Rationale for the Speed-Accuracy Trade-Off. In *2009 IEEE International Conference on Systems, Man and Cybernetics*. 2890–2896. https://doi.org/10.1109/ICSMC.2009.5346580 ISSN: 1062-922X.

[42] Larry Tesler. 2012. A Personal History of Modeless Text Editing and Cut/Copy-Paste. *Interactions* 19, 4 (July 2012), 70–75. https://doi.org/10.1145/2212877.2212896

[43] Daniel Vogel and Patrick Baudisch. 2007. Shift: A Technique for Operating Pen-Based Interfaces Using Touch. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '07)*. ACM, New York, NY, USA, 657–666. https://doi.org/10.1145/1240624.1240727

[44] Feng Wang and Xiangshi Ren. 2009. Empirical Evaluation for Finger Input Properties in Multi-Touch Interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*. Association for Computing Machinery, New York, NY, USA, 1063–1072. https://doi.org/10.1145/1518701.1518864

[45] Haijun Xia, Michael Glueck, Michelle Annett, Michael Wang, and Daniel Wigdor. 2022. Iteratively Designing Gesture Vocabularies: A Survey and Analysis of Best Practices in the HCI Literature. *ACM Transactions on Computer-Human Interaction* 29, 4 (May 2022), 37:1–37:54. https://doi.org/10.1145/3503537

[46] Shumin Zhai and Per Ola Kristensson. 2012. The Word-Gesture Keyboard: Reimagining Keyboard Interaction. *Commun. ACM* 55, 9 (Sept. 2012), 91–101. https://doi.org/10.1145/2330667.2330689

[47] Mingrui Zhang and Jacob O. Wobbrock. 2020. Gedit: Keyboard Gestures for Mobile Text Editing. In *Proceedings of Graphics Interface 2020 (GI 2020)*. Canadian Human-Computer Communications Society / Société canadienne du dialogue humain-machine, Toronto, Ontario, Canada, 470 – 473. https://doi.org/10.20380/GI2020.47 event-place: University of Toronto.

[48] Maozheng Zhao, Wenzhe Cui, I V Ramakrishnan, and Shumin Zhai. 2021. Voice and Touch Based Error-tolerant Multimodal Text Editing and Correction for Smartphones. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '21)*. Association for Computing Machinery, New York, NY, USA, 17.