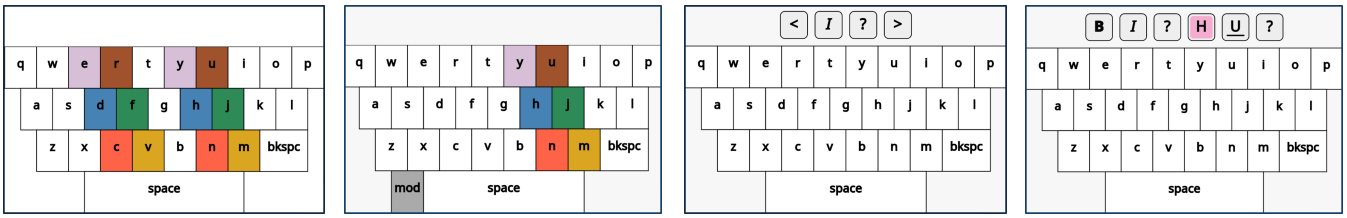
Comparing Chords, Hotkeys, and Toolbars for Mobile Command Selection

Cameron Beattie
University of Saskatchewan
Canada
cameron.beattie@usask.ca

Yen-Ting Yeh
University of Saskatchewan
Canada
allen.yeh@usask.ca

Andy Cockburn
University of Canterbury
New Zealand
andy@cosc.cantebury.ac.nz

Carl Gutwin
University of Saskatchewan
Canada
gutwin@cs.usask.ca



(a) Letter chords (b) SoftCuts (c) Three-layer toolbar (d) One-layer toolbar

Figure 1: Four command entry techniques. (a) Letter chords: commands are selected by pressing two keys that have the same colour. (b) SoftCuts: commands are selected by pressing the modifier key (in grey) and one of the coloured keys. (c) Three-layer toolbar: command are selected by navigating to the correct layer using the < and > buttons, then tapping the command icon. (d) One-layer toolbar: commands are selected by tapping the command icon. ? buttons represent placeholder icons.

Abstract

Users frequently need to enter both text and commands on mobile devices, and these commands can be selected in several ways. Previous research suggests that keyboard shortcuts are substantially faster for command selection than GUI components such as menus or toolbars, but these studies tested desktop rather than mobile environments. It is possible that some of the factors that lead to the advantage of keyboard shortcuts in desktop settings do not have as large an effect in mobile interfaces. To better understand the performance of these mechanisms in mobile typing tasks, we carried out a study (N=120) that compared four command-selection methods: a chording technique, a hotkey technique, and two types of onscreen toolbar (one that required navigation actions, and one in which all commands were visible). Results showed that in both a training game and a phrase-typing task, the toolbar with visible commands was substantially faster than the keyboard-based techniques; however, the toolbar that required navigation actions was slower than all other methods. Our study shows that the performance advantages of keyboard shortcuts do not necessarily translate to the mobile context, but also shows that toolbar performance is strongly determined by the number of navigation actions.

CCS Concepts

• **Human-centered computing** → **Interaction techniques**.

Keywords

Command Selection, Keyboard Shortcuts, Chording, Text Entry

ACM Reference Format:
Cameron Beattie, Yen-Ting Yeh, Andy Cockburn, and Carl Gutwin. 2026. Comparing Chords, Hotkeys, and Toolbars for Mobile Command Selection. In *The Conference on Graphics Interface*. ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/1122445.1122456

1 Introduction

In many mobile systems, people need to select commands while they are entering text. For example, a user might select commands to change the text style, to save the document, or to adjust the zoom level, all interleaved with typing text on the soft keyboard. Traditional design principles from desktop systems suggest that commands can be selected either using GUI controls such as toolbars and menus, or with shortcuts such as gestures or keyboard hotkeys [10, 31, 41]. For mobile devices, which of these strategies is best? Previous studies in HCI have shown that for physical keyboards, selecting commands with hotkeys is substantially faster

GI '26, Waterloo, Canada

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/10.1145/1122445.1122456

than with GUI controls (e.g., [10, 31]). There are two main reasons for this performance advantage. First, if the user's hands are on the keyboard in order to enter text, then it takes time to move one hand to a pointing device such as the mouse [10]. Second, more actions are often required for visual navigation through GUI controls (if the target command is not currently visible [45]) compared to simply pressing the keys of a hotkey.

However, previous studies comparing hotkeys to GUIs have focused almost entirely on desktop environments; and although shortcut techniques have been developed for mobile touchscreen devices (e.g., [1, 25, 26]), these have not been tested against the toolbar-based interfaces that are available in mobile applications (e.g., Figure 2). It is therefore not clear whether shortcuts will also be faster than GUI controls on mobile devices.

The traditional performance advantages of keyboard shortcuts may not be as large for mobile devices. First, operating a GUI control on a mobile device does not require that the user move their hand from the keyboard to a pointing device, since on mobile, both typing and GUI controls are operated with the same digits. Second, the distance to GUI controls is relatively small on mobile systems, particularly when the toolbar is located directly above the keyboard (the most common layout on mobile platforms). Therefore, it is possible that GUI controls could perform better in a mobile context.

To explore this possibility, we carried out a study that compared performance with two types of mobile shortcuts to GUI toolbars. The first mechanism, called Letter Chords [8], uses chorded letter keys to select commands (Figure 1 (a)). The second mechanism, SoftCuts [15, 16], is a direct translation of traditional hotkeys to the mobile context, with a modifier key plus a command key (Figure 1 (b)). We compared these shortcut mechanisms to two kinds of GUI toolbar placed above the keyboard: a three-layer toolbar that requires navigation actions to reach some of the commands (Figure 1 (c)), and a one-layer toolbar in which all commands are visible, so that no navigation is required (Figure 1 (d)).

The Letter Chords technique is novel, and there is little information available about how to design chords for mobile keyboards. Therefore, we first carried out a design study to compare the usability of chords with different horizontal spacing, keyboard row, and side of the keyboard. This study provided a set of guidelines for the development of effective chords for the main performance study.

The main study was a between-participants online experiment (N=120) that included two phases: first, a training game that helped participants learn how to use the chords, the SoftCuts hotkeys, or the toolbars to select text-styling commands; and second, a typing task where participants typed phrases that contained several words with different styles. We measured trial time, command time, and errors for the game phase, and typing speed, command time, and errors for the typing task. Our main research questions were: how the two shortcut techniques compared to the toolbars; which shortcut technique was better; and how the three-layer toolbar compared to the one-layer toolbar.

The main study's results showed that for both the training game and the typing task, the one-layer toolbar was fastest. For the game, command selection time was between 347ms and 459ms faster than with the other mechanisms (27%–36%). For the typing task, both command selection time and typing speed were better with the one-layer toolbar (command selection was 13%–53% faster, and typing speed was 13%–24% higher).

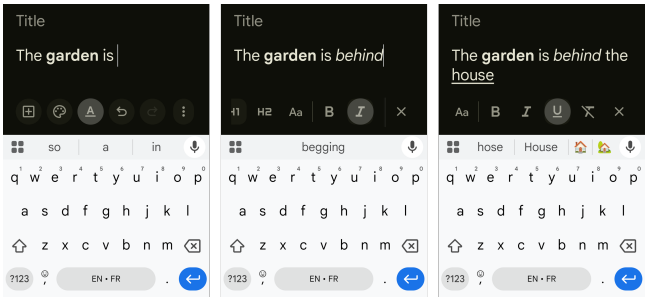


Figure 2: GUI toolbar in Android Notes. Left: to change the text style, the user taps A to view the Styles toolbar. Middle: the user taps I to choose Italics. Right: to choose Underline, the user swipes the toolbar right and taps U.

However, the comparison between one-layer and three-layer toolbars showed that the additional navigation actions of the three-layer toolbar led to substantially slower performance (worse than all of the other techniques). This result suggests that the performance of mobile toolbars is strongly dependent on the number of actions needed to select a command, and if even one navigation action is required before command selection, shortcut mechanisms can be a viable alternative.

Overall, our work provides valuable new information about the design of command selection techniques for mobile text-entry scenarios: we demonstrate that GUI controls such as toolbars can be a fast choice depending on the availability of the command, and that previous results about the performance advantages of keyboard shortcuts do not necessarily translate to the mobile context; however, we also show that mobile designers should carefully consider the number of navigation actions that are needed to select commands from keyboard toolbars.

2 Related Work

There are many ways to issue commands during text entry. In desktop text editors, command entry is typically performed through toolbars and hierarchical menus, which require visual search and navigation, or through hotkeys that rely on memorized key combinations. On mobile devices, command entry in text editing primarily relies on toolbar-based touch controls and contextual interface elements, requiring users to navigate interface components to access editing operations. These approaches differ in the extent to which they rely on visual guidance versus learned recall.

High-performance interaction techniques typically improve command performance through one or more of three strategies: reducing the number of steps required, reducing the time taken per step, or allowing steps to be executed in parallel [7]. Many techniques reduce steps by replacing hierarchical navigation with memory-based retrieval, as seen in keyboard shortcuts, command aliases, and expert-mode systems [7, 8, 22, 38, 47]. Others reduce step time by improving target access with techniques such as command maps, split menus, pop-up menus, and target relocation methods [22, 45, 46]. A third strategy is to parallelize interaction by combining command

selection with action initiation, as in Toolglasses and FlowMenus, or through bimanual and multitouch input techniques [7–9, 23].

2.1 Shortcuts on Mobile Devices

Mobile devices present a different design space for shortcut interaction than desktop systems because of limited screen space, direct touch input, and the absence of persistent peripheral input devices such as keyboards and mice. As a result, many shortcut techniques for mobile focus on preserving expert performance while minimizing occlusion, navigation overhead, and mode switching.

2.1.1 Multi-finger input. Many mobile shortcut techniques use multitouch to expand the command vocabulary beyond single-point touch. For example, FastTap uses thumb-and-finger combinations to invoke a spatially stable grid overlay for rapid command selection on tablets [25]. Finger-count menus similarly use the number of touching fingers to select commands, enabling rapid expert access without traversing hierarchical menus [3, 4]. Other work extends this idea through finger-based 3D gestures, where finger posture and movement encode menu commands [30].

2.1.2 Marking menus. Another major direction adapts gestural shortcuts for mobile devices. Several systems extend Marking Menus to touchscreens by allowing users to issue directional gestures instead of navigating menus [6, 18, 58]. One approach uses a near-surface version of marking menus to support bimanual interaction on mobile devices [24]. These approaches preserve the expert advantages of desktop marking menus while addressing mobile constraints such as finger occlusion and limited display area.

2.1.3 Gestures. Gesture-based techniques provide another way to reduce command-selection overhead on mobile devices by replacing explicit interface navigation with memorized strokes or directional input. Several systems support text editing directly on or near the soft keyboard, including gestural commands for cursor movement, selection, and clipboard operations [19, 57]. Other work has explored elicited mobile command gestures [32], large command vocabularies through gesture recall [29], and shortcut systems inspired by desktop hotkeys [44]. Costagliola et al. similarly proposed a gestural editing technique for touchscreen text editing and found that it outperformed traditional widget-based interaction when text size was large enough to support reliable gesture execution [11]. While gesture techniques can improve efficiency, they often require users to learn and recall gesture vocabularies.

2.2 Hotkeys

Keyboard shortcuts (i.e., hotkeys) are a well-studied technique that can substantially reduce command-selection time. Researchers have studied several aspects of hotkey use, including overall performance gains [31, 48], strategies for encouraging users to switch to hotkeys (e.g., [5, 21, 22, 38, 47]), and adapting hotkeys to other settings such as virtual reality [28] or soft keyboards [15].

One limitation of hotkeys is that they require pressing a modifier key (such as Control), forcing users to move their hand from a natural typing position in order to reach the modifier. Previous work has shown that modifier use can slow command selection, motivating techniques that avoid modifiers (e.g., single command keys without modifiers was shown to be 200ms faster per selection than modifier-based commands [7]). Similarly, CtrlMouse moved the modifier to a mouse button, with users preferring mouse-based modifiers in tasks with more pointing actions [43].

2.2.1 Hotkeys on mobile devices. Mobile devices do not commonly provide physical modifier keys to support hotkeys. Prior work has therefore explored adding modifier functionality to soft keyboards to enable command execution without relying on hierarchical menus or toolbars [14–16]. SoftCuts introduced modifier-based shortcuts on mobile by using multitouch input to activate commands directly from a soft keyboard. Rather than requiring users to memorize shortcuts in advance, SoftCuts reveals available command mappings when the modifier key is activated, allowing users to leverage spatial memory during selection. The technique supports three input methods: user-maintained interaction, in which the modifier key is held while pressing a command key; two sequential taps, where the modifier and command keys are tapped consecutively; and sliding gestures, where users drag from the modifier key to the target command. Among these, two sequential taps produced the best performance. However, the evaluation of SoftCuts did not directly compared it against common mobile methods such as toolbars.

2.3 Chording

Keyboard-based chording involves pressing two or more keys simultaneously to select an input that is different from either of the individual keys. Chording keyboards have been explored since the 1950s [40] and have appeared in many HCI contexts, including the Augment/NLS system [13] and Westerman's work which influenced multitouch input on the iPhone [50].

2.3.1 Chording for text entry. Chording techniques have been developed for both one-handed and two-handed text entry. One-handed devices include Engelbart's five-switch keyboard, capable of 31 states [13], and commercially-available devices such as the Twiddler keyboard [35]. Two-handed chording keyboards, such as stenography systems, allow users to enter phonemes rather than individual letters on a 22-key layout, achieving speeds up to 375 words per minute [52]. Chording has also been applied to mobile devices. On mobile keypads, chording methods achieved comparable choice reaction times and error rates relative to conventional single-key input [34]. Concurrent chording techniques for mobile text entry significantly outperformed variations of MultiTap [51]. Wu and colleagues explored mobile chording keyboards with multiple characters per key, introducing techniques such as MagArea and MemoryTap [55, 56], showing improvements in speed and errors.

2.3.2 Chording for command selection. Chording is also a high-performance mechanism for command selection. On traditional keyboards, modifier-based hotkeys are a form of chording, while on touchscreens, multi-finger techniques such as Finger-Count Menus [4] and FastTap [25] use simultaneous inputs to select commands.

Au and colleagues' DownChords and UpChords proposed using chords of 2-4 letter keys to select commands, and designed variants that worked on both key-down events (DownChords) and key-up events (UpChords) [2]. Their studies showed that key-overlap timings could be used to reliably differentiate chords from regular typing. Multi-finger chording has been applied to tablets, with

structured chord-command mappings supporting short- and long-term retention [49]. FingerArc and FingerChord leverage hand posture recognition to enable secondary finger movements for related shortcut selection [59].

More recently, Beattie et al. showed that desktop command entry can be accelerated by mapping commands to two-key Letter Chords on standard keyboards, avoiding modifier keys while preserving typing posture. In this technique, commands are issued by pressing two standard keys simultaneously, with chord assignments derived through a keyboard design study. Letter Chords were shown to be 200ms faster than hotkeys in tasks with intermixed command execution and typing [8]. This technique offers another memory-based approach to command entry by allowing users to press multiple keys simultaneously rather than sequentially like hotkeys.

Ghomi et al. examined the biomechanics of touchscreen chords, finding that users prefer relaxed chords with fewer fingers and avoid patterns that require lifting one finger while adjacent fingers remain in contact [20]. Based on these insights, they proposed biomechanically informed chord-design guidelines and introduced Arpège, a dynamic system providing feedforward and feedback to support chord learning.

2.4 Toolbars

Research on toolbar design has explored how layout and interaction mechanics affect efficiency and usability. Michalski et al. examined user preferences for vertical graphical toolbars, analyzing how panel location, configuration, and target size influenced task efficiency and satisfaction [39]. Lee et al. proposed a "tool ball" mechanism operated by the mouse wheel, which reduced on-screen icon clutter, expanded mouse-wheel functionality, and accelerated task performance [33].

Other work has examined how toolbar adaptation influences feature awareness and discoverability. Findlater et al. investigated personalized toolbars, showing that while personalization can improve efficiency for familiar tasks, it may reduce overall feature awareness and negatively affect performance on unfamiliar tasks [17]. Similarly, Wogalter et al. found that although icons contribute to GUI usability, multiple other interface factors affect how users locate and interpret commands [54].

Research has also explored higher-level command organization across interfaces and devices. Darejeh et al. studied ribbon interfaces, using two usability evaluations to iteratively improve visual and cognitive aspects of grouped toolbar design [12]. Majrashi et al. examined multi-device menu interfaces, finding that consistent menu item order across devices improves learning efficiency and reduces workload, while layout differences have little effect on post-transition performance [37]. Together, these studies show that toolbar design involves balancing efficiency, discoverability, cognitive load, and consistency across interaction contexts.

3 Letter Chords Design Study

Although implementation details for mobile toolbars and hotkeys have been examined in prior work (e.g., [15, 16]), there is little information available to guide the design of keyboard-based chording techniques for command selection (and the limited previous work has focused on desktop devices [2, 8]). To provide this understanding, we carried out a crowd-sourced study that tested user speed and accuracy with several types of chords on mobile devices. Participants completed the study on their own mobile web browser. See Figure 3 for the design study interface.

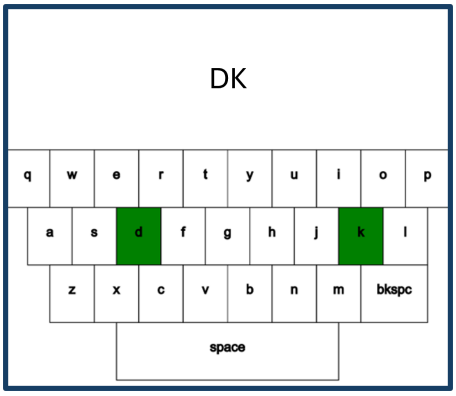


Figure 3: Interface used in Letter Chords design study. The chord to enter was displayed at top centre, and the required chord keys were coloured green on the custom soft keyboard. Non-chord key presses were coloured grey.

3.1 Design Study: Letter Chord Dimensions

We limited our study to two-letter chords, as most users type on mobile devices with two thumbs [27, 42]. We identified three dimensions along which two-letter chords vary and tested representative chords for each (66 total):

- *Horizontal spacing*. The number of keys between letters (e.g., j k has spacing 0, while j l has spacing 1). We tested spacings from 0–8.
- *Chord row*. Chords can use keys on the same row (e.g., t u) or different rows (e.g., r j, u n).
- *Keyboard side*. We tested chords on the left side (e.g., a d), right side (e.g., j p), and across both sides (e.g., e i). Although input was unconstrained, we assumed most chords would be entered with two thumbs.

All tested chords and their dimensions are provided in the supplementary material.

3.2 Design Study: chord entry task

Participants entered each of the test chords multiple times, over three blocks. For each trial, the system showed the target letter chord and also coloured the required keys on the mobile keyboard (Figure 3). This method of cueing the task focuses on the physical execution time for entering the letter chord, and does not involve any learning or retrieval of the chord from memory.

Participants were instructed to press the indicated letters simultaneously, as quickly and accurately as possible. A chord was considered successful when the required keys were pressed at the same time. Chord order was randomized before each block, and incorrect keys were shown in grey. Trials timed out after five seconds.

After the task, participants rated the difficulty of each chord on a 7-point scale and could provide optional comments.

3.3 Participants, Procedure, and Design

Thirty participants (12 men, 18 women; mean age 36.6) were recruited through Prolific. Participation was restricted to adults with a task approval rate of at least 90% to improve data quality; survey responses were also checked for inattentive behaviour. No participants were excluded.

After consent, demographics, and instructions, participants completed the chord-entry task and rated the difficulty of each chord. Ethical approval was granted by the University of Saskatchewan.

The study used a within-participants exploratory design to investigate the differences between various letter chords in the categories described above. Factors and levels were: *Horizontal spacing* (0 to 8); *Chord row* (top, middle, bottom, top-middle, top-bottom, middle-bottom); and *Side of Keyboard* (left, right, both). Dependent variables included performance measures (completion time and incorrect keys) and subjective ratings of difficulty for each letter chord. Participants completed 198 trials (66 letter chords × 3 blocks); with 30 participants, there were 5,940 trials observed in total.

3.4 Design Study: Results

The following sections report on each of the chord dimensions introduced above. Charts show mean completion times, mean error rates (based on the number of incorrect keys pressed), and mean participant difficulty ratings; error bars show 95% confidence intervals across all trials. All letter chords tested are provided in the supplementary materials.

3.4.1 Horizontal Spacing. Figure 4 shows results for chords with horizontal spacing from 0 (adjacent keys) to 8 (opposite sides of the keyboard). Zero-spacing chords (e.g., g h) had high completion times, error rates, and timeout rates because users' thumbs often obstructed each other, making these chords difficult to enter. Due to these skewed results, subsequent analyses exclude zero-spacing chords. Overall, chords with spacing of 2–4 keys had the fastest completion times, while zero- and one-key spacing produced the highest error rates and worst user rating.

3.4.2 Chord Row. Figure 5 compares chords across row combinations. Completion times were similar across rows, ranging from 913ms (middle-bottom) to 1017ms (top-middle). Same-row chords produced higher error rates and worse ratings than mixed-row chords, although this may partly reflect the inclusion of one-space chords.

3.4.3 Side of Keyboard. Figure 6 compares chords on the left side, right side, and both sides of the keyboard. Chords using both sides had the fastest completion times, lowest difficulty ratings, and low error rates. Left-side chords also performed well, while right-side chords were substantially slower, produced more than twice as many errors, and received the worst ratings.

3.4.4 Summary. Overall, the design study provides useful guidelines for letter chords on mobile devices: chords should have a spacing of 2, 3, or 4 keys; chords can use any combination of rows; and chords should use either both sides of the keyboard, or the left side. We used these results to inform the design of the Letter Chords condition in the main study (described next) comparing several command-selection techniques.

4 Performance Study of Command Techniques

The goal of the second study was to assess the performance of four command-selection techniques (Letter Chords, SoftCuts, a Three-Layer toolbar, and a One-Layer toolbar) for typing scenarios on mobile devices. The study included two phases: a training game in which participants practised four text-styling commands with one of the command techniques; and a phrase-typing task in which participants typed text and entered styling commands to match a target phrase. The study was implemented as a P5JS web application that presented trials and logged participant actions (see Figures 7).

4.1 Command-Selection Techniques

The techniques described below were used to select four styling commands (Bold, Italic, Underline, Highlight) in the two phases of the study (summarized in Table 1).

- *Letter Chords.* Style commands were issued by pressing two-letter chords, with letter combinations chosen based on the design guidelines described above. Four chord pairs were assigned to four style commands, with two additional chord pairs as placeholder commands to match the number of commands in the Three-Layer Toolbar. The chord keys were visually highlighted on the soft keyboard. See Figure 1 (a).
- *SoftCuts.* The SoftCuts technique was implemented as described in previous work [16]. We chose to implement the user-maintained input technique because it best matches desktop hotkeys. The mobile keyboard included a modifier key mod that enabled hotkey-style shortcuts similar to desktop hotkeys. When the modifier key was pressed and held, six style keys were highlighted on the keyboard: four keys issued style commands, and two additional keys served as placeholders to match the number of commands in the Three-Layer Toolbar. Pressing these keys when holding the mod key issued a command. See Figure 1 (b).
- *One-Layer Toolbar.* Six icons were displayed on a toolbar located directly above the soft keyboard; all icons were visible at all times. Four icons issued style commands, while two additional icons served as placeholders (shown as ?) to match the number of commands in the Three-Layer Toolbar. Participants could tap an icon to turn on or turn off a style. See Figure 1 (d).
- *Three-Layer Toolbar.* Six icons were distributed across three toolbar layers. Layer 1 contained the icon for Bold **B**, a placeholder icon, and a navigate-right icon >. Layer 2 contained a navigate-left icon <, an icon for Italics *I*, a placeholder icon, and a navigate-right icon. Layer 3 contained a navigate-left icon, the Underline icon U, and the Highlight icon H. Placeholder icons were included in this condition to equalize the number of command icons per level. See Figure 1 (c).

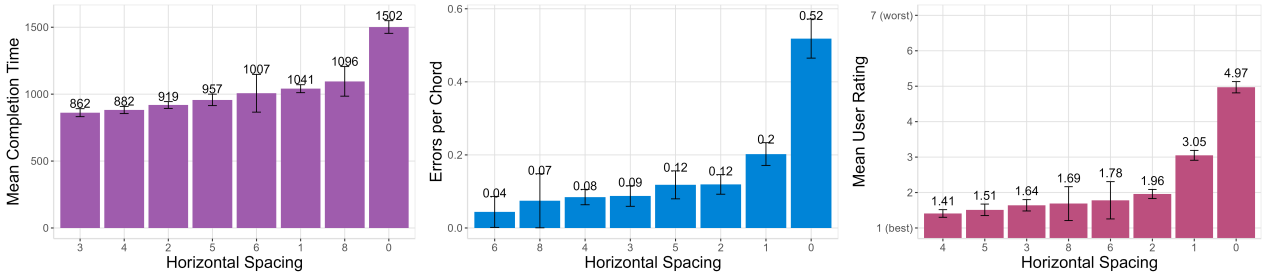


Figure 4: Horizontal Spacing. Left: completion time. Middle: error rate. Right: user difficulty rating.

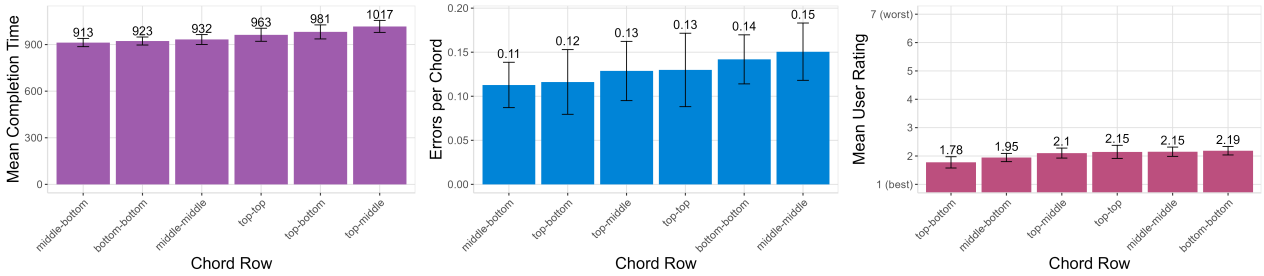


Figure 5: Chord Row Combination. Left: completion time. Middle: error rate. Right: user difficulty rating.

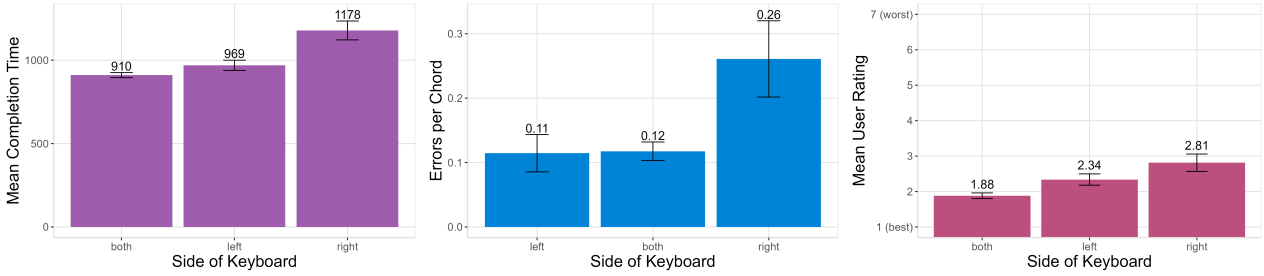


Figure 6: Side of Keyboard. Left: completion time. Middle: error rate. Right: user difficulty rating.

Technique	Bold (Blue)	Italic (Green)	Underline (Orange)	Highlight (Pink)
1-Layer Tool.	B	I	U	H
3-Layer Tool.	B L1	I L2	U L3	H L3
Letter Chords	D H	F J	C N	E Y
SoftCuts	mod H	mod J	mod N	mod Y

Table 1: Performance study: Command Mechanisms.

4.2 Phase 1: Training game

We built a simple game in which styled words fell from the top of the screen, and the participant had to enter the style command and type the word before the word passed the bottom of the screen. In the example shown in Figure 7 (left), a participant would first enter the command for Highlight (by pressing e and y) and then type the word. The speed of the falling words was set so that the word was on screen for five seconds regardless of display size.

The game randomly chose words from a predefined list of three-letter English words, and randomly applied one of five formats (the four styles noted above, as well as plain text). Participants were required to complete 120 styled trials to complete the training.

4.3 Phase 2: Phrase-typing task

After they had successfully completed the game, and after taking a short break, participants moved to a phrase-typing task in which they typed phrases where some of the words had a style, using the same four styles as in the game (Figure 7). Participants had to both turn the style on (before typing the word) and off (after typing the word). Our phrase set had phrases with the following characteristics: 16-20 characters (mean 18.48), 5 words per phrase, and 2-5 characters per word (mean 3.7) [36].

To ensure a similar distribution of navigation levels in the three-layer toolbar, styles were assigned as follows: 20 zero-navigation commands (0-Nav), 17 one-navigation commands (1-Nav), and 23

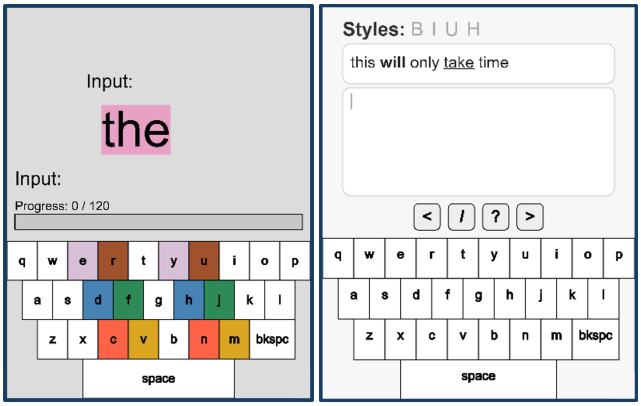


Figure 7: Example study interfaces. Left: game UI with Letter Chords. Right: phrase-typing UI with Three-Layer Toolbar.

two-navigation commands (2-Nav). The same phrase set and style assignments were used across all techniques.

Turning a style off in the three-layer toolbar was always a 0-navigation selection. Participants were not required to turn off the style for the final word of a phrase, resulting in 50 0-Nav style-off commands. Participants typed 25 phrases; feedback about what the user had typed was shown immediately below the target phrase.

4.4 Procedure and Participants

The study system presented the styling game, phrase-typing tasks, and questionnaires, and recorded all study data. Participants first completed informed consent and demographics forms, and were assigned to a between-participants technique group (either Letter Chords, SoftCuts, One-Layer Toolbar or Three-Layer Toolbar); they were then shown an instruction page that explained the game and phrase-typing tasks. They then completed the two parts of the study. Following the phrase-typing tasks, participants completed a questionnaire for that condition based on the NASA-TLX, but with two additional questions asking about the technique's accuracy and speed. Ethical approval was provided by the ethics board of the University of Saskatchewan.

120 new participants (47 men, 72 women, 1 undeclared, mean age 33.2) were recruited through Prolific; participation was again restricted to people over 18, and required a 90% or higher task acceptance rate. No participants were removed from the analysis for answering the same way on all questionnaire questions.

4.5 Study Design

The study used a between-participants factorial design with factors *Technique* (One-Layer Toolbar, Three-Layer Toolbar, Letter Chords, SoftCuts) and *Block* (1-5). The two phases (Game and Phrase Typing) were analysed separately. In addition, to better understand the effects of the number of navigation actions in the Three-Layer Toolbar condition, we carried out additional analyses comparing the three levels of navigation (0-Nav, 1-Nav, and 2-Nav) to the other conditions. This allowed us to compare the keyboard-based techniques to toolbars with different amounts of navigation.

Dependent variables for the game phase were trial time, command selection time, and subjective ratings from the post-task questionnaire. For the phrase-typing task, measures were typing speed, command selection time, and subjective ratings. There were 30 participants in each condition (120 total), with each completing 120 styled words in the game and 25 phrases in the typing task.

4.6 Performance Study Results

Analyses are organized below by study phase and dependent variable. Effect size for significant ANOVA results is reported as generalized eta squared (η^2). Completion time data for the different techniques did not have equal variance (Levene's test $p < .05$), so ANOVA and follow-up tests use log-transformed data; follow-up tests were corrected using the Holm-Bonferroni method.

Game trials with a trial time over 9000 ms (based on pilot tests to determine a reasonable range for task completion) were removed from the dataset, as were trials with trial times over 3 standard deviations from the mean, in each technique and block combination (1047 of 16335 game trials were removed – 6.4%, 755 of 17647 commands in the game were removed – 4.3%). In the phrase task, we removed trials where typing speed was outside of 3 standard deviations from the mean in each technique and block combination (12 of 2997 phrase trials were removed – 0.4%, and 392 out of 16289 commands – 2.4%). In all charts below, error bars show 95% confidence intervals; in all violin plots, tails of the violins are trimmed using ggplot defaults. Most charts show both the overall mean time for the Three-Layer Toolbar, as well as means for each amount of navigation (0-Nav, 1-Nav, 2-Nav). ANOVA tables show the results of our primary analyses (using the four levels of factor Technique), but for significant main effects, we also include pairwise comparisons between the individual navigation levels of Three-Layer Toolbar and the other conditions.

4.6.1 Game Phase. **Trial time** for the game measured the time to issue the style command type the word, including the time to correct any errors. Figure 8 summarizes the trial time results, and Table 2 (top) summarizes findings from the two-way ANOVA, revealing significant effects of Technique and Block, as well as a significant interaction. The main findings are that the One-Layer Toolbar (2412 ms) and the 0-Nav trials with the Three-Layer Toolbar (2484 ms) were fastest, showing significantly lower times than Letter Chords (2887 ms), SoftCuts (3136 ms), and the Three-Layer Toolbar when one (3390 ms) or two (3246 ms) navigation actions were required. Letter Chords was also significantly faster than both 1-Nav and 2-Nav levels of the Three-Layer Toolbar.

Command time was measured as the time from the end of the previous action to the selection of the next correct command; this included navigation time in the Three-Layer Toolbar. Figures 9 and 10 summarize the mean command time for each condition (including separate representations for the overall mean of the Three-Layer Toolbar as well as individual means for 0-Nav, 1-Nav, and 2-Nav trials) Table 2 (middle) presents the results of a two-way ANOVA, revealing significant effects of Technique and Block, as well as a significant interaction. As shown in Figure 10, the main difference over the five blocks of the study is that performance with Letter Chords was worse to begin with, but increased more quickly than the other techniques.

Figure 8: Game Mean Trial Time ± 95% CI, by Technique (numbers inside violins show mean time in ms).

The main findings are that One-Layer Toolbars (command time of 1265 ms), and the 0-Nav trials in the Three-Layer Toolbar (1287 ms) were the fastest techniques, showing significantly lower times than Letter Chords (1686 ms), SoftCuts (1612 ms), and the Three-Layer Toolbar when one (1979 ms) or two (1906 ms) navigation steps were required. Letter Chords and SoftCuts were also significantly faster than both 1-Nav and 2-Nav trials in the Three-Layer Toolbar.

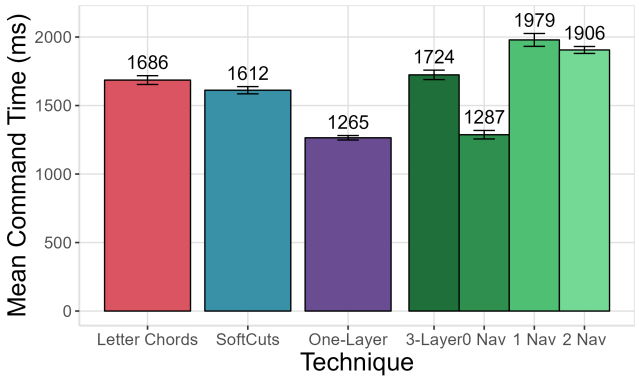


Figure 9: Game Command Time ± 95% CI, by Technique.

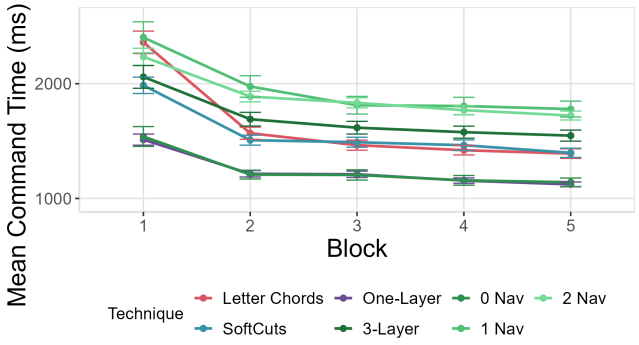


Figure 10: Game Command Time ± 95% CI, by Block.

Errors were measured as the number of times participants selected an incorrect style per trial (but did not count regular typing mistakes). Figure 11 shows the number of errors per trial, and Table 2 (bottom) summarizes findings from the two-way ANOVA, revealing significant effects of Technique and Block, as well as a significant interaction. Overall, Letter Chords (0.02) had significantly fewer errors than both the One-Layer Toolbar (0.05) and SoftCuts (0.08). The Three-Layer Toolbar (0.02) also had fewer errors than SoftCuts, and the 1-Nav and 2-Nav levels of the Three-Layer Toolbar had fewer errors than the One-Layer Toolbar.

4.6.2 Phrase-Typing Task. **Typing speed** in characters per second (CPS) was measured as the number of correct characters typed divided by the total trial time (including time to correct errors). Figure 12 summarizes the typing speed results, and Table 3 (top) summarizes findings from the two-way ANOVA, revealing significant effects of Technique and Block, as well as a significant interaction. Pairwise analysis showed that the One-Layer Toolbar (1.24 CPS) had higher speed than the Three-Layer Toolbar (1.0 CPS). Letter Chords (1.1 CPS) and SoftCuts (1.06 CPS) were slightly faster than Three-Layer, although these differences were not significant.

Command time for the phrase task was again measured as the time between the end of the previous action and the selection of the next required command. Figures 13 and 14 summarize these results, and Table 3 (middle) summarizes findings from the two-way ANOVA, revealing significant effects of Technique and Block, as well as a significant interaction.

Pairwise tests showed that the One-Layer Toolbar (1218 ms), Letter Chords (1373 ms), SoftCuts (1538 ms) and 0-Nav commands in the Three-Layer Toolbar (1349 ms) were faster than 1-Nav (2098 ms) or 2-Nav (2157 ms) commands in the Three-Layer Toolbar. The One-Layer Toolbar was also significantly faster than SoftCuts.

We also examined differences between commands used to turn a style *On* and those used to turn the style *Off*. Figure 15 shows these results (note that in the Three-Layer Toolbar, all *Off* commands required no navigation since they were already visible). Performance was similar across *On* and *Off* actions for Letter Chords and the One-Layer Toolbar; but for SoftCuts, *On* commands were slower than *Off* commands.

Errors were measured as the rate of extra commands issued per required command. For this analysis, we cannot separate the

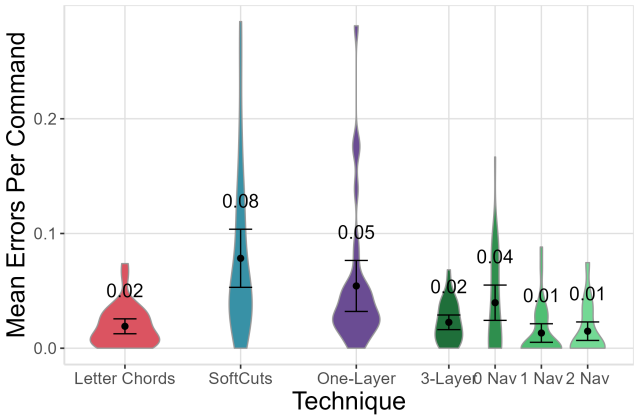


Figure 11: Game Errors/Command ± 95% CI, by Technique.

Game Trial Time				
Factor	**DF (n,d)**	**F**	**p**	η^2
Technique	3, 116	1.82e+01	**9.12e-10**	0.29
Block	4, 696	2.73e+02	**2.41e-120**	0.26
Techn. × Block	12, 464	4.23	**2.33e-06**	0.016
Pairwise Contrasts for Technique: 1-Layer < Letter Chords, SoftCuts, 3-Layer, p<.005				
Pairwise Contrasts split by required navigation: 0 Nav, 1-Layer < Letter Chords, SoftCuts, 1 Nav, 2 Nav, p<.005 Letter Chords < 1 Nav, 2 Nav, p<.05				
Game Command Time				
Factor	**DF (n,d)**	**F**	**p**	η^2
Technique	3, 116	3.09e+01	**9.61e-15**	0.36
Block	4, 464	1.27e+02	**5.74e-73**	0.24
Techn. × Block	20, 464	6.23	**3.14e-10**	0.04
Pairwise Contrasts for Technique: Letter Chords, SoftCuts, 1-Layer < 3-Layer, p<.05 Letter Chords, SoftCuts < 1-Layer, p<.005				
Pairwise Contrasts split by required navigation: 1-Layer, 0-Nav < Letter Chords, SoftCuts, 1-Nav, 2-Nav, p<.001 Letter Chords, SoftCuts < 1-Nav, 2-Nav, p<.001				
Game Command Errors				
Factor	**DF (n,d)**	**F**	**p**	η^2
Technique	3, 116	12.02	**6.57e-07**	0.09
Block	4, 464	86.6	**6.54e-55**	0.33
Techn. × Block	12, 464	8.27	**3.34e-14**	0.13
Pairwise Contrasts for Technique: 3-Layer, Letter Chords < SoftCuts, 1-Layer, p<.05				
Pairwise Contrasts split by required navigation: Letter Chords < 1-Layer, SoftCuts, p<.001 0-Nav, 1-Nav, 2-Nav < SoftCuts, p<.001 1-Nav, 2-Nav < 1-Layer, p<.005				

Table 2: Factorial Results – Game Phase. Top: trial time. Middle: command time. Bottom: errors. Main analyses use the four levels of Technique; we also include additional pairwise contrasts that include the different navigation amounts of Three-Layer Toolbar.

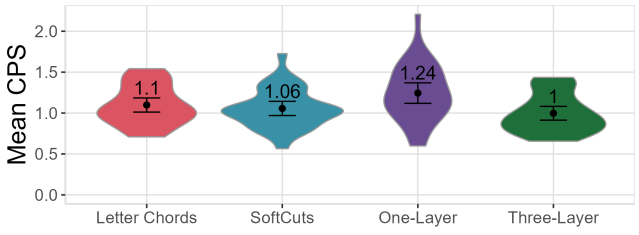


Figure 12: Mean Typing Speed ± 95% CI, by Technique (numbers inside violins show characters per second).

Three-Layer Toolbar into navigation steps, so only the overall mean is shown in the summary of Figure 16. Table 3 (bottom) summarizes findings from the two-way ANOVA, revealing only a significant effect of Block.

4.6.3 Subjective Measures: Speed, Accuracy, Effort, and Preferences. At the end of the study, participants completed the NASA-TLX

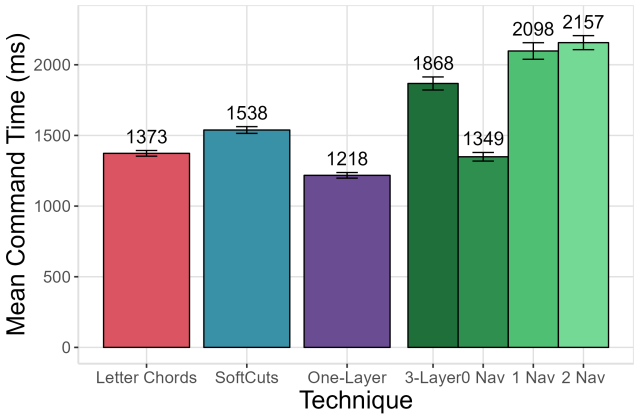


Figure 13: Phrase Command Time ± 95% CI, by Technique.

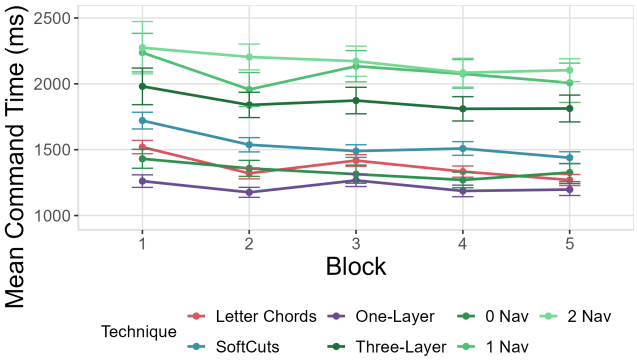


Figure 14: Phrase Command Time by Block.

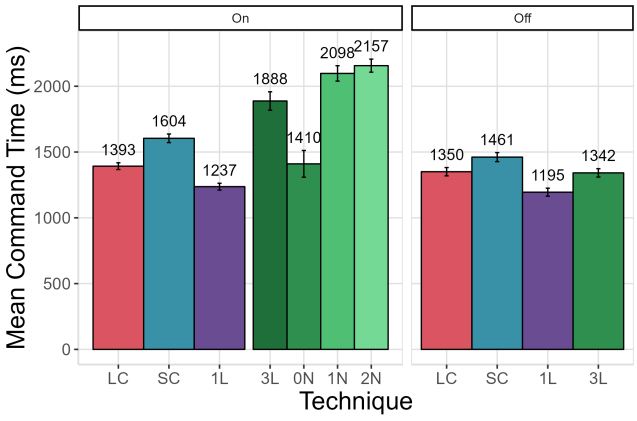


Figure 15: Phrase command time: style-on vs. style-off.

survey and additional questions about speed and accuracy for their assigned technique. For all questions, we applied the Aligned Rank Transform [53] and carried out one-way ANOVAs on each question. Mean responses are shown in Figure 17 – these were similar across the conditions, with a significant effect of technique observed only

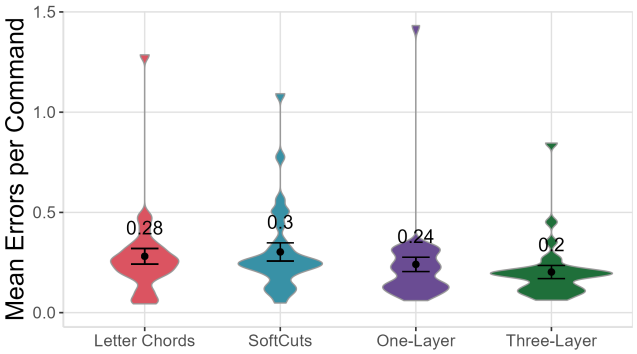


Figure 16: Phrase Mean Errors ± 95% CI, by Technique.

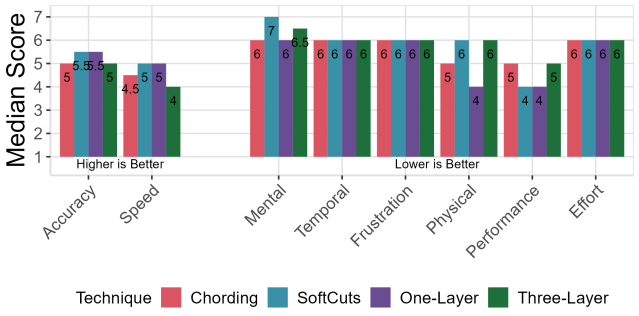


Figure 17: Subjective Responses. Left: Perceived accuracy and speed. Right: NASA-TLX.

Typing Speed				
Factor	**DF (n,d)**	**F**	**p**	η^2
Technique	3, 116	3.26	**2.42e-02**	0.06
Block	4, 464	11.19	**1.11e-08**	0.02
Techn. × Block	12, 464	1.10	3.60e-01	6.9e-03
Pairwise Contrasts for Technique: 1-Layer > 3-Layer, p<.05				
Command Time				
Factor	**DF (n,d)**	**F**	**p**	η^2
Technique	3, 116	10.03	**6.25e-06**	0.19
Block	4, 464	9.57	**1.92e-07**	0.01
Techn. × Block	12, 464	5.55	**6.68e-09**	0.017
Pairwise Contrasts for Technique: Letter Chords, 1-Layer < 3-Layer, p<.05 1-Layer < SoftCuts, p<.005				
Pairwise Contrasts split by required navigation: 1-Layer < Softcuts, p<.005 Letter Chords, 1-Layer, Softcuts, 0 Nav < 1 Nav, 2 Nav, p<.001				
Command Errors				
Factor	**DF (n,d)**	**F**	**p**	η^2
Technique	3, 116	2.12	0.102	0.02
Block	4, 464	5.39	**3.01e-04**	0.025
Techn. × Block	12, 464	1.01	0.44	0.015

Table 3: Factorial Results – Phrase Task. Top: typing speed in CPS. Middle: command time. Bottom: errors. Main analyses use the four levels of Technique, but for Command Time, we also include additional pairwise contrasts that include the different navigation amounts of Three-Layer Toolbar.

for Physical Demand ($F_{3,116} = 3.11, p = .029$). On this scale, the SoftCuts and Three-Layer Toolbar conditions had higher physical demand ratings (median 6) than One-Layer Toolbar (4).

5 Discussion

Our studies provide several novel findings about command selection mechanisms for mobile devices. For both the training game and the typing task, the One-Layer Toolbar was fastest. In the game, One-Layer command selection time was 421ms faster than with Letter Chords (33%), 347ms faster than with SoftCuts (27%), and 459ms faster than with the Three-Layer Toolbar (36%). In the phrase-typing task, both command selection time and typing speed were better with the One-Layer Toolbar: command selection was 13% faster than with Letter Chords, 30% faster than with SoftCuts, and 53% faster than with the Three-Layer Toolbar; typing speed was 13% higher than with Letter Chords, 17% higher than with SoftCuts, and 24% higher than with the Three-Layer Toolbar.

The advantage for the One-Layer Toolbar, however, did not extend to the toolbar which required navigation to reach the commands. Selections from the Three-Layer Toolbar that required either one or two navigation actions were significantly slower than all of the other techniques in both the game and the phrase-typing task: for example, command times in the phrase task for 0-Nav and 1-Nav actions were more than 500ms slower than either LetterChords or SoftCuts, and more than 700ms slower than either the 0-Nav actions or the One-Layer Toolbar.

5.1 Explanation for main results

5.1.1 Why was the One-Layer Toolbar faster? There are several possible reasons for the strong performance of the One-Layer Toolbar. First, as suggested earlier, it may be that the limitations of toolbars that exist in desktop environments are not as prevalent for mobile interfaces: in particular, there was no requirement to switch the hands from typing to GUI interaction (since on mobiles, the thumbs are used for both modes), and the distance required to reach the toolbar was low (and within reach of the natural range of motion for the users' thumbs).

Second, users may now be highly familiar with toolbar-based commands during typing, given that these toolbars exist in many mobile applications. In contrast, SoftCuts was a new technique for users (even though all participants were familiar with modifier-based hotkeys from desktop keyboards), and no participants were familiar with letter-based chording. Even though the game phase of the study provided some degree of equalization across the conditions (participants completed 120 commands and so became reasonably comfortable with the techniques) but there could still have been a familiarity effect in favour of the toolbar.

Third, toolbars also provide additional benefits over keyboard-based command mechanisms: they provide visual guidance rather than requiring memory retrieval, and if retrieval was not automatic for participants in the Letter Chords and SoftCuts conditions, this may have provided an advantage for the toolbar; in addition, the One-Layer Toolbar also only requires a single action, whereas

both keyboard techniques required two actions (although for Letter Chords, these could be carried out in parallel). Multiple actions may have required more mental preparation time.

Fourth, SoftCuts required that users press a modifier key, which has been shown in previous studies to add time to the action [7].

5.1.2 Why were Three-Layer Toolbars slow? The performance of the Three-Layer Toolbar was strongly related to the number of navigation actions required to reach the command. When the required command was already visible in the toolbar (i.e., a 0-Nav selection), then it could be chosen as quickly as the One-Layer Toolbar. However, when a command required navigation actions, it was substantially slower.

The main reason for this reduction in performance is the cost of (a) recognizing that the desired command is not visible in the toolbar, and (b) performing one or more navigation actions to move to the correct part of the toolbar. Even after users became familiar with the layout of the toolbar (e.g., the later blocks shown in Figures 14 and 10), the additional navigation actions required substantial extra time. Therefore, designers must carefully consider the number of navigation actions in a toolbar.

This clear relationship between the number of navigation actions and the performance of the toolbar designs raises questions of how many commands can be shown on a real-world mobile toolbar, and how often navigation will be required. The answers to these questions depend on the size of the device – but as an example, the above-keyboard toolbar for the Google Docs app on a Pixel 8 XL device shows six formatting commands and two menus with additional commands (eight icons total). If eight commands can be shown on the toolbar, it is likely that the front layer of the toolbar can cover a large proportion of the commands needed for a typing task (the Google Docs toolbar has 16 commands in total, so half would be available on the front layer of a toolbar). It is important to note that although one navigation action led to substantially reduced performance, having navigation for some commands did not slow down selections for the visible commands (i.e., 0-Nav selections were reliably fast). If the eight most frequent commands were shown on the first layer of a toolbar, the majority of high-use commands could be selected quickly.

However, if navigation actions are required by the toolbar's design (e.g., some apps require a navigation action to get to any of the formatting commands), then it is possible that Letter Chords and SoftCuts could be higher-performing options. Single-layer toolbars are ultimately constrained by the number of commands an application can expose before additional layers or navigation are required. In contrast, SoftCuts and Letter Chords do not inherently impose this limitation, though they rely more heavily on memory than visual feedback.

5.2 Generalizing the Findings

Running the study on Prolific meant that the study systems were run by users with a wide range of experience, and were run on a wide range of mobile phones with different screen sizes. The real-world variation in the sample, and the fact that people used their own devices, increases the external validity of our results – for example, the study saw 25 different types of phones, with several different screen sizes and resolutions.

Our tasks included a high proportion of commands to the amount of typing and used only text-styling commands; real-world tasks will have a much wider variation in the proportion of commands to typing, and could involve several other types of commands. Despite this difference, we believe that our findings will translate to real typing-based tasks, because the basic mechanisms used for each technique in our studies are the same as what would be seen in a real-world task. However, the proportion of commands that are needed in real-world tasks varies greatly, and this will affect the real-world differences between the techniques.

Overall, the main lesson for designers of mobile systems is that the current standard approach for command selection (a toolbar placed immediately above the keyboard) can provide higher performance than keyboard-based techniques – with the caveat that the design of the toolbar must be carefully considered to avoid unnecessary navigation actions. When toolbars require users to navigate between layers or menus to access commands, performance decreases substantially, making them slower than our shortcut-based techniques. In these situations, the reduced interaction overhead of Letter Chords and SoftCuts can provide a performance advantage.

5.3 Limitations and Future Work

There are several limitations to what we could explore in our studies, and each of these leads to opportunities for further research. First, participants learned and used only four commands in our studies (with two additional placeholder commands that were not used). We used a small number of commands due to restrictions on the amount of training time available in the study, but many real-world applications will use more commands, and so further studies will be needed to examine learning and performance with larger command sets. Second, although our phrase task realistically combined commands with typing, the tasks were still artificial (and involved more style commands than what would be used in most real-world typing scenarios). Further studies are needed to examine the performance of command-selection mechanisms in more realistic typing tasks. Third, our keyboard implementation was simplified and may not fully reflect real-world smartphone keyboards, which could influence typing performance and user behavior. Features such as autocorrect and gesture support may affect interaction with our techniques. Further studies using operating system keyboards are needed to better understand real-world usability.

6 Conclusion

In mobile applications where users need to enter both text and commands, there are several ways that users can select commands: using GUI controls such as toolbars, or using shortcuts such as hotkeys or chords. Previous studies suggest that keyboard shortcuts are substantially faster for command selection than GUI components, but this research focused on desktop contexts rather than mobile environments. To better understand the performance of different command-selection mechanisms in mobile typing tasks, we carried out a study that compared two shortcut techniques (Letter Chords and SoftCuts) with two toolbar techniques (one that required navigation actions, and one in which all commands were visible). The study showed that in both the training game and the phrase-typing task, the One-Layer Toolbar was substantially faster

than either of the keyboard-based techniques; however, the Three-Layer Toolbar that required navigation actions was slower than all other methods. Our study shows that the performance advantages of hotkeys that have been suggested by previous research do not necessarily translate to the mobile context, but also indicates that toolbar performance is strongly dependent on the number of navigation actions. If applications include a large number of commands, which can make multi-layer or navigational toolbars unavoidable, shortcut-based techniques can become faster alternatives to toolbar-based command selection. Overall, our work provides valuable new information about the design of command selection techniques for mobile text entry.

References

[1] Ahmed Sabbir Arif, Michel Pahud, Ken Hinckley, and Bill Buxton. 2020. Experimental study of stroke shortcuts for a touchscreen keyboard with gesture-redundant keys removed. In *Graphics Interface 2014*. AK Peters/CRC Press, 43–50.

[2] Oscar Kin-Chung Au, Kira Yip-Wo Yeung, and Jacky Kit Cheung. 2016. Down-Chord and UpChord: A New Style of Keyboard Shortcuts based on Simultaneous Key-down and Key-up Events. In *Proceedings of the Fourth International Symposium of Chinese CHI* (San Jose, USA) *(Chinese CHI '16)*. Association for Computing Machinery, New York, NY, USA, Article 3, 10 pages. doi:10.1145/2948708.2948715

[3] Gilles Bailly, Eric Lecolinet, and Yves Guiard. 2010. Finger-count & radial-stroke shortcuts: 2 techniques for augmenting linear menus on multi-touch surfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Atlanta, Georgia, USA) *(CHI '10)*. Association for Computing Machinery, New York, NY, USA, 591–594. doi:10.1145/1753326.1753414

[4] Gilles Bailly, Jörg Müller, and Eric Lecolinet. 2012. Design and evaluation of finger-count interaction: Combining multitouch gestures and menus. *International Journal of Human-Computer Studies* 70, 10 (2012), 673–689. doi:10.1016/j.ijhcs.2012.05.006 Special issue on Developing, Evaluating and Deploying Multi-touch Systems.

[5] Gilles Bailly, Thomas Pietrzak, Jonathan Deber, and Daniel J. Wigdor. 2013. Métamorphe: augmenting hotkey usage with actuated keys. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Paris, France) *(CHI '13)*. Association for Computing Machinery, New York, NY, USA, 563–572. doi:10.1145/2470654.2470734

[6] Jens Bauer, Achim Ebert, Oliver Kreylos, and Bernd Hamann. 2013. Marking Menus for Eyes-Free Interaction Using Smart Phones and Tablets. In *Availability, Reliability, and Security in Information Systems and HCI*, Alfredo Cuzzocrea, Christian Kittl, Dimitris E. Simos, Edgar Weippl, and Lida Xu (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 481–494.

[7] Cameron Beattie, Carl Gutwin, Andy Cockburn, and Eric Redekopp. 2025. Understanding and Improving the Performance of Action Pointing. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (CHI '25)*. Association for Computing Machinery, New York, NY, USA, Article 856, 23 pages. doi:10.1145/3706598.3713761

[8] Cameron Beattie, Yen-Ting Yeh, Andy Cockburn, and Carl Gutwin. 2026. Investigating the Design and Performance of Letter Chords: Alternative Command Entry For Typing. In *Proceedings of the 2026 International Conference on Advanced Visual Interfaces (AVI 2026)*. doi:10.1145/3811427.3811430 In press.

[9] Eric A Bier, Maureen C Stone, Ken Fishkin, William Buxton, and Thomas Baudel. 1994. A taxonomy of see-through tools. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 358–364.

[10] Stuart K Card, Thomas P Moran, and Allen Newell. 1980. The keystroke-level model for user performance time with interactive systems. *Commun. ACM* 23, 7 (1980), 396–410.

[11] Gennaro Costagliola, Mattia De Rosa, and Vittorio Fuccella. 2018. A technique for improving text editing on touchscreen devices. *Journal of Visual Languages Computing* 47 (2018), 1–8. doi:10.1016/j.jvlc.2018.04.002

[12] Ali Darejeh and Dalbir Singh. 2014. An investigation on Ribbon interface design guidelines for people with less computer literacy. *Computer Standards Interfaces* 36, 5 (2014), 808–820. doi:10.1016/j.csi.2014.01.006

[13] D Engelbart and WK English. 1968. A research centre for augmenting human intellect. In *Proceedings of the Fall 1968 AFIPS Computer Conference 33*. 395–410.

[14] Katherine Fennedy and Hyowon Lee. 2019. Multitouch Keyboard Revisited: Enhancing Moded Interaction Through Redesigning Structure and Switching Techniques. In *Proceedings of the 5th International ACM In-Cooperation HCI and UX Conference* (Jakarta, Surabaya, Bali, Indonesia) *(CHIuXiD'19)*. Association for Computing Machinery, New York, NY, USA, 36–45. doi:10.1145/3328243.3328249

[15] Katherine Fennedy, Sylvain Malacria, Hyowon Lee, and Simon Tangi Perrault. 2020. Investigating Performance and Usage of Input Methods for Soft Keyboard Hotkeys. In *22nd International Conference on Human-Computer Interaction with Mobile Devices and Services* (Oldenburg, Germany) *(MobileHCI '20)*. Association for Computing Machinery, New York, NY, USA, Article 29, 12 pages. doi:10.1145/3379503.3403552

[16] Katherine Fennedy, Angad Srivastava, Sylvain Malacria, and Simon T. Perrault. 2022. Towards a Unified and Efficient Command Selection Mechanism for Touch-Based Devices Using Soft Keyboard Hotkeys. *ACM Trans. Comput.-Hum. Interact.* 29, 1, Article 4 (Jan. 2022), 39 pages. doi:10.1145/3476510

[17] Leah Findlater and Joanna McGrenere. 2010. Beyond performance: Feature awareness in personalized interfaces. *International Journal of Human-Computer Studies* 68, 3 (2010), 121–137. doi:10.1016/j.ijhcs.2009.10.002

[18] Jeremie Francone, Gilles Bailly, Laurence Nigay, and Eric Lecolinet. 2009. Wavelet menus: a stacking metaphor for adapting marking menus to mobile devices. In *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services* (Bonn, Germany) *(MobileHCI '09)*. Association for Computing Machinery, New York, NY, USA, Article 49, 4 pages. doi:10.1145/1613858.1613919

[19] 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* (Paris, France) *(CHI '13)*. Association for Computing Machinery, New York, NY, USA, 2785–2794. doi:10.1145/2470654.2481385

[20] Emilien Ghomi, Stéphane Huot, Olivier Bau, Michel Beaudouin-Lafon, and Wendy Mackay. 2013. Arpège: Learning Multitouch Chord Gestures Vocabularies. *ITS 2013 - Proceedings of the 2013 ACM International Conference on Interactive Tabletops and Surfaces*, 209–218. doi:10.1145/2512349.2512795

[21] Emmanouil Giannisakis, Gilles Bailly, Sylvain Malacria, and Fanny Chevalier. 2017. IconHK: Using Toolbar button Icons to Communicate Keyboard Shortcuts. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. Association for Computing Machinery, New York, NY, USA, 4715–4726. doi:10.1145/3025453.3025595

[22] Tovi Grossman, Pierre Dragicevic, and Ravin Balakrishnan. 2007. Strategies for accelerating on-line learning of hotkeys. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '07)*. Association for Computing Machinery, New York, NY, USA, 1591–1600. doi:10.1145/1240624.1240865

[23] François Guimbretière, Andrew Martin, and Terry Winograd. 2005. Benefits of merging command selection and direct manipulation. *ACM Trans. Comput.-Hum. Interact.* 12, 3 (Sept. 2005), 460–476. doi:10.1145/1096737.1096742

[24] François Guimbretière and Chau Nguyen. 2012. Bimanual marking menu for near surface interactions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Austin, Texas, USA) *(CHI '12)*. Association for Computing Machinery, New York, NY, USA, 825–828. doi:10.1145/2207676.2208521

[25] Carl Gutwin, Andy Cockburn, Joey Scarr, Sylvain Malacria, and Scott C. Olson. 2014. Faster command selection on tablets with FastTap. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Toronto, Ontario, Canada) *(CHI '14)*. Association for Computing Machinery, New York, NY, USA, 2617–2626. doi:10.1145/2556288.2557136

[26] Ken Hinckley, Koji Yatani, Michel Pahud, Nicole Coddington, Jenny Rodenhouse, Andy Wilson, Hrvoje Benko, and Bill Buxton. 2010. Pen + touch = new tools. In *Proceedings of the 23nd Annual ACM Symposium on User Interface Software and Technology* (New York, New York, USA) *(UIST '10)*. Association for Computing Machinery, New York, NY, USA, 27–36. doi:10.1145/1866029.1866036

[27] Xinhui Jiang, Yang Li, Jussi P.P. Jokinen, Viet Ba Hirvola, Antti Oulasvirta, and Xiangshi Ren. 2020. How We Type: Eye and Finger Movement Strategies in Mobile Typing. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–14. doi:10.1145/3313831.3376711

[28] Min Joo Kim, Yu Gyeong Son, Yong Min Kim, and Donggun Park. 2025. Comparing typing methods for uppercase input in virtual reality: Modifier Key vs. alternative approaches. *International Journal of Human-Computer Studies* 193 (2025), 103385. doi:10.1016/j.ijhcs.2024.103385

[29] Per Ola Kristensson and Shumin Zhai. 2007. Command strokes with and without preview: using pen gestures on keyboard for command selection. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '07)*. Association for Computing Machinery, New York, NY, USA, 1137–1146. doi:10.1145/1240624.1240797

[30] Arun Kulshreshth and Joseph J. LaViola. 2014. Exploring the usefulness of finger-based 3D gesture menu selection. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Toronto, Ontario, Canada) *(CHI '14)*. Association for Computing Machinery, New York, NY, USA, 1093–1102. doi:10.1145/2556288.2557122

[31] David M. Lane, H. Albert Napier, S. Camille Peres, and Aniko Sandor. 2005. Hidden Costs of Graphical User Interfaces: Failure to Make the Transition from Menus and Icon Toolbars to Keyboard Shortcuts. *International Journal of Human–Computer Interaction* 18, 2 (2005), 133–144. arXiv:https://doi.org/10.1207/s15327590ijhc1802_1 doi:10.1207/s15327590ijhc1802_1

[32] 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 Trans. Comput.-Hum. Interact.* 27, 5, Article 33 (Aug. 2020), 38 pages. doi:10.1145/3396233

[33] Kuo-Wei Lee and Ying-Chu Lee. 2012. Design and validation of an improved graphical user interface with the 'Tool ball'. *Applied Ergonomics* 43, 1 (2012), 57–68. doi:10.1016/j.apergo.2011.03.004

[34] Seongil Lee and Sang Hyuk Hong. 2004. Chording as a Text Entry Method in Mobile Phones. In *Mobile Human-Computer Interaction - MobileHCI 2004*, Stephen Brewster and Mark Dunlop (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 456–460.

[35] Kent Lyons, Thad Starner, Daniel Plaisted, James Fusia, Amanda Lyons, Aaron Drew, and E. W. Looney. 2004. Twiddler typing: one-handed chording text entry for mobile phones. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vienna, Austria) *(CHI '04)*. Association for Computing Machinery, New York, NY, USA, 671–678. doi:10.1145/985692.985777

[36] I. Scott MacKenzie and R. William Soukoreff. 2003. Phrase sets for evaluating text entry techniques. In *CHI '03 Extended Abstracts on Human Factors in Computing Systems* (Ft. Lauderdale, Florida, USA) *(CHI EA '03)*. Association for Computing Machinery, New York, NY, USA, 754–755. doi:10.1145/765891.765971

[37] Khalid Majrashi. 2019. Post-transitioning user performance on cross-device menu interfaces. *International Journal of Human-Computer Studies* 130 (2019), 130–151. doi:10.1016/j.ijhcs.2019.06.001

[38] Sylvain Malacria, Gilles Bailly, Joel Harrison, Andy Cockburn, and Carl Gutwin. 2013. Promoting Hotkey use through rehearsal with ExposeHK. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Paris, France) *(CHI '13)*. Association for Computing Machinery, New York, NY, USA, 573–582. doi:10.1145/2470654.2470735

[39] Rafał Michalski. 2011. Examining users' preferences towards vertical graphical toolbars in simple search and point tasks. *Computers in Human Behavior* 27, 6 (2011), 2308–2321. doi:10.1016/j.chb.2011.07.010

[40] J Noyes. 1983. Chord keyboards. *Applied Ergonomics* 14, 1 (1983), 55–59.

[41] Halla Olafsdottir and Caroline Appert. 2014. Multi-touch gestures for discrete and continuous control. In *Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces* (Como, Italy) *(AVI '14)*. Association for Computing Machinery, New York, NY, USA, 177–184. doi:10.1145/2598153.2598169

[42] Kseniia Palin, Anna Maria Feit, Sunjun Kim, Per Ola Kristensson, and Antti Oulasvirta. 2019. How do People Type on Mobile Devices? Observations from a Study with 37,000 Volunteers. In *Proceedings of the 21st International Conference on Human-Computer Interaction with Mobile Devices and Services* (Taipei, Taiwan) *(MobileHCI '19)*. Association for Computing Machinery, New York, NY, USA, Article 9, 12 pages. doi:10.1145/3338286.3340120

[43] Thomas Pietrzak, Sylvain Malacria, and Gilles Bailly. 2014. CtrlMouse et TouchCtrl: duplicating mode delimiters on the mouse. In *Proceedings of the 26th Conference on l'Interaction Homme-Machine* (Villeneuve d'Ascq, France) *(IHM '14)*. Association for Computing Machinery, New York, NY, USA, 38–47. doi:10.1145/2670444.2670447

[44] Gulnar Rakhmetulla, Yuan Ren, and Ahmed Sabbir Arif. 2023. GeShort: One-Handed Mobile Text Editing and Formatting with Gestural Shortcuts and a Floating Clipboard. *Proc. ACM Hum.-Comput. Interact.* 7, MHCI, Article 212 (Sept. 2023), 23 pages. doi:10.1145/3604259

[45] Joey Scarr, Andy Cockburn, Carl Gutwin, and Andrea Bunt. 2012. Improving command selection with CommandMaps. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Austin, Texas, USA) *(CHI '12)*. Association for Computing Machinery, New York, NY, USA, 257–266. doi:10.1145/2207676.2207713

[46] Joey Scarr, Andy Cockburn, Carl Gutwin, Andrea Bunt, and Jared E. Cechanowicz. 2014. The usability of CommandMaps in realistic tasks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Toronto, Ontario, Canada) *(CHI '14)*. Association for Computing Machinery, New York, NY, USA, 2241–2250. doi:10.1145/2556288.2556976

[47] Joey Scarr, Andy Cockburn, Carl Gutwin, and Philip Quinn. 2011. Dips and ceilings: understanding and supporting transitions to expertise in user interfaces. In *Proceedings of the sigchi conference on human factors in computing systems*. 2741–2750.

[48] Yilei Shi, Haimo Zhang, Hasitha Rajapakse, Nuwan Tharaka Perera, Tomás Vega Gálvez, and Suranga Nanayakkara. 2018. GestAKey: Touch Interaction on Individual Keycaps. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–12. doi:10.1145/3173574.3174170

[49] Julie Wagner, Eric Lecolinet, and Ted Selker. 2014. Multi-finger chords for hand-held tablets: recognizable and memorable. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Toronto, Ontario, Canada) *(CHI '14)*. Association for Computing Machinery, New York, NY, USA, 2883–2892. doi:10.1145/2556288.2556958

[50] Wayne Westerman. 2004. System and method for recognizing touch typing under limited tactile feedback conditions. https://patents.google.com/patent/US6677932B1 US Patent 6,677,932.

[51] Daniel Wigdor and Ravin Balakrishnan. 2004. A comparison of consecutive and concurrent input text entry techniques for mobile phones. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vienna, Austria) *(CHI '04)*. Association for Computing Machinery, New York, NY, USA, 81–88. doi:10.1145/985692.985703

[52] Wikipedia. 2025. *Stenotype*. Retrieved September 9, 2025 from https://en.wikipedia.org/wiki/Stenotype

[53] Jacob O. Wobbrock, Leah Findlater, Darren Gergle, and James J. Higgins. 2011. The aligned rank transform for nonparametric factorial analyses using only anova procedures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vancouver, BC, Canada) *(CHI '11)*. Association for Computing Machinery, New York, NY, USA, 143–146. doi:10.1145/1978942.1978963

[54] Michael S Wogalter. 2009. Usability problems in word processing applications. *Proceedings of the XVIIth Triennial International Ergonomics Association* (2009).

[55] Fong-Gong Wu, Yu-Chun Huang, and Meng-Long Wu. 2014. New chording text entry methods combining physical and virtual buttons on a mobile phone. *Applied Ergonomics* 45, 4 (2014), 825–832. doi:10.1016/j.apergo.2013.10.011

[56] Fong-Gong Wu and Wen-Zhou Shi. 2018. The input efficiency of chord keyboards. *International Journal of Occupational Safety and Ergonomics* 24, 4 (2018), 638–645. arXiv:https://doi.org/10.1080/10803548.2017.1362171 doi:10.1080/10803548.2017.1362171 PMID: 28849996.

[57] Mingrui Zhang and Jacob O. Wobbrock. 2020. Gedit: Keyboard Gestures for Mobile Text Editing. In *Graphics Interface 2020*. https://openreview.net/forum?id=CZ938F7zVF

[58] Jingjie Zheng, Xiaojun Bi, Kun Li, Yang Li, and Shumin Zhai. 2018. M3 Gesture Menu: Design and Experimental Analyses of Marking Menus for Touchscreen Mobile Interaction. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–14. doi:10.1145/3173574.3173823

[59] Jingjie Zheng, Blaine Lewis, Jeff Avery, and Daniel Vogel. 2018. FingerArc and FingerChord: Supporting Novice to Expert Transitions with Guided Finger-Aware Shortcuts. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology* (Berlin, Germany) *(UIST '18)*. Association for Computing Machinery, New York, NY, USA, 347–363. doi:10.1145/3242587.3242589