TapStr: A Tap and Stroke Reduced-Qwerty for Smartphones

Mohammad Akbor Sharif
Human-Computer Interaction Group
University of California, Merced
Merced, California, United States
msharif@ucmerced.edu

Gulnar Rakhmetulla
Human-Computer Interaction Group
University of California, Merced
Merced, California, United States
grakhmetulla@ucmerced.edu

Ahmed Sabbir Arif
Human-Computer Interaction Group
University of California, Merced
Merced, California, United States
asarif@ucmerced.edu

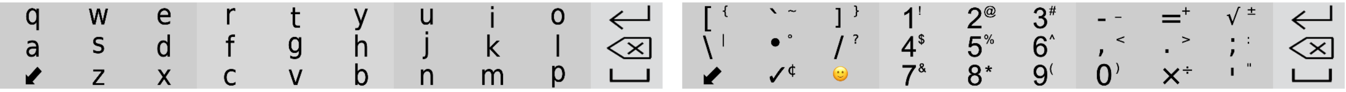


Figure 1: TapStr is an unambiguous reduced-Qwerty that maps the standard Qwerty in a single row divided into four zones. Tapping on a zone enters the central character and stroking on a zone towards one of the eight characters around the border enters the respective character. Dwelling for 500 ms upon a tap or stroke enters the uppercase letters or the secondary characters, distinguished using a smaller font (right). Dwelling for 1000 ms activates or deactivates Caps Lock. Stroking towards '↙' switches the layout for numeric and special characters (right) and stroking towards '☺' switches the layout for emojis.

ABSTRACT

TapStr is a single-row reduced-Qwerty that enables text entry on smartphones by performing taps and directional strokes. It is an unambiguous keyboard, thus does not rely on a statistical decoder to function. It supports the entry of uppercase and lowercase letters, numeric characters, symbols, and emojis. Its purpose is to provide an economic alternative to virtual Qwerty when saving touchscreen real-estate is more desired than the entry speed and a convenient alternative when users do not want to repeatedly switch between multiple layouts for mixed-case alphanumeric text and symbols (such as a password). In a short-term study, TapStr yielded about 11 WPM with plain phrases and 8 WPM with phrases containing uppercase letters, numbers, and symbols.

CCS CONCEPTS

- **Human-centered computing → Text input**; **Gestural input**.

KEYWORDS

Text entry; reduced keyboard; virtual keyboard

ACM Reference Format:
Mohammad Akbor Sharif, Gulnar Rakhmetulla, and Ahmed Sabbir Arif. 2020. TapStr: A Tap and Stroke Reduced-Qwerty for Smartphones. In *Companion Proceedings of the 2020 Conference on Interactive Surfaces and Spaces (ISS '20 Companion), November 8–11, 2020, Virtual Event, Portugal.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3380867.3426208

Virtual Qwerty has become the dominant method for text entry on smartphones. It is evident that with enough practice, mobile users can reach a competitive entry speed with virtual Qwerty [12]. However, one problem with virtual Qwerty is that it occupies about 40% of a smartphone display. Saving smartphone real-estate is crucial since cluttered display affects user performance [11]. Besides, the saved space could be used to display more of what has been entered so far to keep the user aware of the context of a chat and the flow of a passage, which improves both writing speed [10] and quality [22]. Besides, entering special characters, symbols, and emojis is difficult with virtual Qwerty (especially for novices) since it requires users to repeatedly switch between multiple layouts (typically, four or more) for the intended characters. TapStr is a novel single-row 65×8 mm reduced-Qwerty that occupies only about 7% of a smartphone display. It uses only three layouts to enable the entry of alphanumeric and a range of special characters and emojis by performing taps and directional strokes on the keyboard. The motivation of TapStr is not to replace the conventional virtual Qwerty but to provide an economic alternative when saving touchscreen real-state is more desired than the entry speed (for example, when users do not want to repeatedly scroll up and down to see the email they are responding to or the video they are commenting on) and a convenient alternative when users do not want to switch back and forth between different layouts to enter mixed-case alphanumeric text and symbols (for example, when entering a password or URL).

1 TAPSTR KEYBOARD

TapStr maps the standard Qwerty layout in a single row divided into four zones. The zones are distinguished using different sheds of grey (Fig. 1). TapStr uses the Qwerty layout to facilitate skill transfer [7, 17], but occupies only 5.24 cm^2 of a 74.5 cm^2 smartphone screen, when a virtual Qwerty occupies 23–29 cm^2, freeing up 25–32% of the display. The first three 19.66×8 mm zones contain nine characters each: one in the middle, four along the four sides, and four in the four corners. Users tap on a key to enter its central character and stroke towards the characters around the border to enter the respective characters (Fig. 1). A stroke can be initiated anywhere on the zone containing the target character and end anywhere on the touchscreen, even outside the keyboard. TapStr registers all touch-point movements over 10 pixels as strokes and determines its direction based on the start and end points. This

---

ISS '20 Companion, November 8–11, 2020, Virtual Event, Portugal

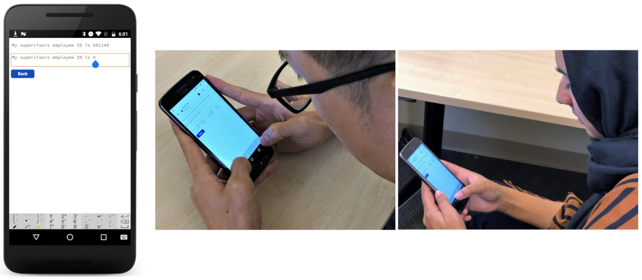
ACM ISBN 978-1-4503-7526-9/20/11.
https://doi.org/10.1145/3380867.3426208

enables users to change the direction of a stroke midway when they mistakenly stroked towards an incorrect character. For uppercase letters, users first tap or stroke for the intended letter, then dwell for 500 ms. If a dwell is activated by mistake, users can slightly shift the finger to disable it. The keyboard provides visual feedback on dwell by flipping the layout from lowercase to uppercase. The layout returns to lowercase upon touch-up. Users can dwell for 1000 ms to enable (or disable) Caps Lock. The fourth 6×8 mm zone is for Enter, Backspace, and Space, which are entered by an up-stroke, tap, and down-stroke, respectively. Stroking towards '↙' switches the layout for numeric and special characters and vice versa (Fig. 1). TapStr places all paired symbols side-by-side and groups similar symbols together (e.g., places all mathematical operators on the third zone) to facilitate mnemonic strategies. The layout categorizes the most frequent symbols as *primary* characters and the least frequent ones as *secondary* characters[1]. Primary and secondary characters are visually distinguished in the layout using a bigger and a smaller font-size (Fig. 1, right), respectively. Symbols are entered using the same mechanism as letters: users tap or stoke to enter the primary characters and dwell to enter the secondary characters. Dwelling provides visual feedback by flipping the font-size of the primary and the secondary characters. The '☺' symbol switches the layout for emojis. However, we disable this feature in the study. We also implemented a predictive system for the keyboard to enable word prediction and auto-correction. This feature was also disabled during the study to eliminate a penitential confounding factor.

2 RELATED WORK

There are several reduced keyboards available for mobile devices. Stick Keyboard [8] maps the four rows of Qwerty onto the home row. Although designed for mobile devices, it was evaluated on a desktop computer with a physical prototype, where it yielded 10.4 WPM. 1Line Keyboard [14] is a similar virtual keyboard for tablets. With the support of a statistical decoder, it yielded 30.7 WPM in a longitudinal study. TenGO [21] maps Qwerty onto six keys and uses four extra keys for Space, Backspace, Shift, and to cycle through suggested words. This keyboard was not evaluated in a user study. Gueorguieva et al. [9] designed a virtual keyboard to enable text entry using Morse code. In a longitudinal study, it reached about 7 WPM. Senorita [18] is a virtual chorded keyboard that arranges all letters on eight keys laid out in a single row. In a longitudinal study, it reached 14 WPM. Arif et al. [3] replaced the Space, Backspace, Shift, and Enter keys of a virtual Qwerty with directional strokes to make room for numeric and special characters in the main layout. In a study, it yielded 17.4 WPM. TaS [19] arranges all letters in a 4×2 grid alphabetically. Each key contains one character in the middle and four characters along the four sides. Similarly, TapFlick [15] maps Qwerty onto three keys in a single row. Each key contains nine characters: one in the middle, four along the four sides, and the remaining four in the four corners. Like TapStr, both TaS and TapFlick use taps and directional strokes for text entry. However, these keyboards occupy about the same screen real-estate as conventional virtual Qwerty and do not fully support the entry of numeric and special characters. Besides, some of these keyboards reviewed here are ambiguous, thus rely on aggressive statistical decoders, which makes entering out-of-vocabulary words very difficult, seldom impossible.

Recently, there has been a growing interest in miniature keyboards for smartwatches [1]. The most relevant to our work is SwipeKey [20] that arranges all letters of the English alphabet in a 4×2 grid in an alphabetical order. Each key includes four characters in the four sides, which are entered by performing directional strokes. However, this keyboard does not support the entry of numeric characters and special symbols.



My supervisors employee ID is 881148

My supervisors employee ID is 8

Figure 2: The device and the app used in the study (left) and two participants taking part in the study (right).

3 EXPERIMENT

We conducted a study to compare the performance of TapStr with Gboard, the default Google Android keyboard [13].

3.1 Participants

Twelve participants aged 19–28 years (M = 21.5, SD = 3.1) took part in the study (Fig. 2). Four of them were female and eight were male. They all were proficient in English and experienced virtual Qwerty users (M = 8 years of experience, SD = 2.04). Ten of them were right-handed and two were left-handed. They all received US $10 for volunteering.

3.2 Apparatus

The study used a Motorola Moto G^5 Plus smartphone (150.2×74×7.7 mm, 155 g) running on Android OS 7.0 Nougat at 1080×1920 pixels. All predictive features of the two keyboards were disabled to eliminate a potential confound. Text entry performance was recorded using WebTEM[2], a freely available web application [2].

3.3 Design

The study used a within-subjects design. The independent variables were *keyboard* and *phrase*, and the dependent variables were the performance metrics. We recorded the standard words per minute (WPM) and error rate (%) metrics. Words per minute is the average number of words entered in one minute, where a "word" is measured as 5 characters [4]. Error rate is the average percentage (%) of incorrect characters remained in the transcribed text. The study

---

[1] Due to the absence of a frequency table for symbols, we studied the state-of-the-art virtual keyboards and assumed that all symbols that are placed on a higher layout (which require fewer actions to access) are the most frequently used symbols and the ones that are placed in lower layouts are the least frequently used symbols.

[2] WebTEM: A web application to record text entry metrics, https://WebTEM.site

used two phrase sets. *Plain phrases* are from the MacKenzie and Soukoreff set [16] that are moderate in length (M = 29), contain a few uppercase letters but no numeric or special characters. *Mixed phrases*[3] are from a different set [3] that are also moderate in length (M = 37) but contain on average 7% uppercase letters, 10% numeric characters, and 7% symbols. An example phrase from this set is *“$6.52 is way too much for a bottle of water!”*. Hence, the design was:

12 participants ×
2 keyboards (Qwerty and TapStr, counterbalanced) ×
2 phrase sets (plain and mixed, counterbalanced) ×
15 phrases = 720 phrases, in total.

3.4 Procedure

The study took place in a quiet room. Upon arrival, we demonstrated TapStr and explained the study procedure to all participants. We then collected their consents. We enabled participants to practice with TapStr by entering free-form text for about five minutes. We did not ask them to practice with the Gboard since they all were experienced users of virtual Qwerty. We then started the main study that required participants to transcribe fifteen phrases from two different sets (plain and mixed) using two different keyboards (TapStr and Qwerty). The phrase sets and the keyboards were counterbalanced to eliminate the effect of learning. In each condition, random phrases were presented on the top of the screen, one at a time (Fig. 2). We instructed participants to read the phrases carefully, transcribe them as *fast and accurate* as possible, then press Enter to see the next phrase. Error correction was encouraged but not forced. Logging started after entering the first character and ended with the last. We informed participants that they could take breaks between the conditions or before they start typing a phrase. Upon completion of the study, they completed a short questionnaire that asked them to rate various aspects of the new keyboard.

4 RESULTS

For statistical tests, we removed all phrases that were missing more than ten characters (9% of the data). We used a repeated-measures ANOVA for all analysis since the data did not violate the normality or the sphericity assumptions.

4.1 Entry Speed

An ANOVA identified a significant effect of keyboard ($F_{1,11} = 11.49$, $p < .01$) and phrase ($F_{1,11} = 13.40$, $p < .01$) on entry speed. There was no significant effect of block ($F_{2,22} = 1.86$, $p = .18$) but the keyboard × block ($F_{2,22} = 4.00$, $p < .05$) and the keyboard × phrase × block ($F_{2,22} = 5.70$, $p < .05$) interaction effects were significant. However, the keyboard × phrase ($F_{1,11} = 0.65$, $p = .44$) interaction effect was not significant. Fig. 3 illustrates average entry speed of the two keyboards with plain and mixed phrases.

4.2 Error Rate

An ANOVA failed to identify a significant effect of keyboard ($F_{1,11} = 0.55$, $p = .48$), phrase ($F_{1,11} = 1.48$, $p = .25$), or block ($F_{2,22} = 0.87$, $p = .43$) on error rate. The keyboard × block ($F_{2,22} = 0.77$, $p = .48$), the keyboard × phrase ($F_{1,11} = 1.64$, $p = .25$), and the keyboard × phrase × block ($F_{2,22} = 0.00$, $p = .99$) interaction effects were also not significant. Fig. 4 illustrates average error rate of the two keyboards with plain and mixed phrases.

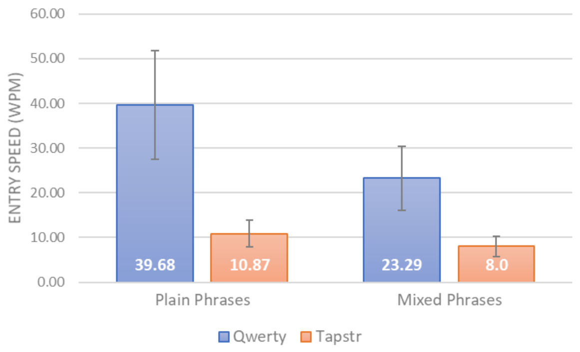


Figure 3: Average entry speed of the two keyboards with plain and mixed phrases. Error bars represent ±1 standard deviation (SD).

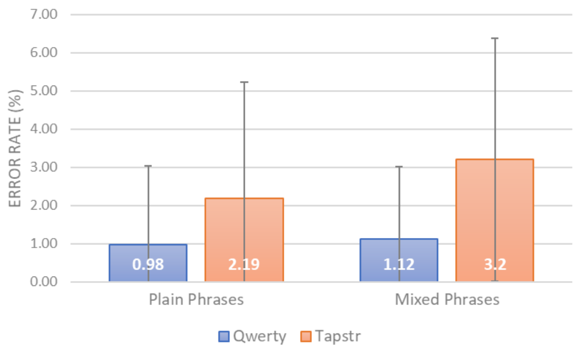


Figure 4: Average error rate (%) of the two keyboards with plain and mixed phrases. Error bars represent ±1 standard deviation (SD).

5 DISCUSSION

TapStr yielded a significantly slower entry speed (71.72%) than Qwerty. We anticipated this since participants were new to the layout, thus needed time to learn it. Besides, TapStr requires performing linear strokes to enter most characters, which most probably contributed towards the slower entry speed. Prior research showed that stroke lengths reduce with practice, as users get more familiar with a new layout [3]. Hence, it is possible that the keyboard will yield a much better entry speed in a longitudinal study. The 500 ms dwell time for the uppercase letters and some special characters may have contributed towards the slower entry speed as well. A prior work attempted to mitigate this by gradually reducing the dwell time as users get more familiar with the system [6]. Further investigation is needed to find out whether this approach benefits TapStr. Entry speed with both keyboards were much slower for

---

[3] Mixed phrases, https://www.asarif.com/pub/mixedset.txt

mixed phrases compared to plain phrases. This is also expected as entering numbers and symbols requires extra effort with both keyboards. But interestingly, entry speed with TapStr dropped at a much lower rate (26%) compared to Qwerty (41%), which indirectly suggests that entering numbers and symbols with TapStr does not require as much effort as Qwerty. Significant interaction effects on entry speed suggest that entry speed during the study, particularly for mixed phrases, improved with practice. A Tukey-Kramer test confirmed that entry speed for mixed phrases was much faster in the last block compared to the first block ($p < .05$). Besides, with both Qwerty ($R^2 = 0.94$) and TapStr ($R^2 = 0.88$), entry speed over the blocks for mixed phrases correlated well to the power law of practice [5].

There was no significant main or interaction effects on error rate. TapStr yielded a much higher error rate (2.69%) than Qwerty (1.05%). A deeper analysis revealed that text entry with TapStr became more accurate with practice ($R^2 = 0.86$), while no such trends were observed with Qwerty. This is likely because participants were already familiar with Qwerty, thus did not trigger as many hit-and-miss as TapStr. Further, participants were becoming more efficient in entering numbers and symbols with both keyboards (Qwerty: $R^2 = 0.85$, TapStr: $R^2 = 0.99$), requiring fewer corrective actions, which correlate well to the power law of practice [5].

Questionnaire data revealed that participants were not very enthusiastic about TapStr. Roughly half of them were against using TapStr for both plain (42%, N = 5) and mixed phrases (50%, N = 6), while the remaining were neutral about it. This demonstrates experienced virtual Qwerty users' reliance and confidence in the keyboard they are familiar with. Further investigation is needed to find out if their initial impression of the keyboard changes after using it for a longer period of time. Participants were mostly neutral about the learnability of the new keyboard. About 25% of them (N = 3) found the keyboard difficult to master, the remaining (N = 9) were neutral.

6 CONCLUSION AND FUTURE WORK

This paper presented TapStr, an unambiguous reduced-Qwerty that occupies only about 7% of a stock smartphone screen. It enables the entry of uppercase and lowercase letters, numeric characters, symbols, and emojis by taps and directional strokes. In a short-term study, TapStr yielded on average 11 WPM entry speed for plain phrases and 8 WPM for mixed phrases with only 2–3% error rate without the support of a statistical decoder or a predictive system. In the future, we will conduct a longitudinal study to investigate if the performance and user preference of the keyboard improve with practice. We will also explore the potential of the keyboard on smaller devices like smartwatches.

REFERENCES

[1] Ahmed Sabbir Arif and Ali Mazalek. 2016. A Survey of Text Entry Techniques for Smartwatches. In *Human-Computer Interaction. Interaction Platforms and Techniques (Lecture Notes in Computer Science)*, Masaaki Kurosu (Ed.). Springer International Publishing, Cham, 255–267. https://doi.org/10.1007/978-3-319-39516-6_24

[2] Ahmed Sabbir Arif and Ali Mazalek. 2016. WebTEM: A Web Application to Record Text Entry Metrics. In *Proceedings of the 2016 ACM International Conference on Interactive Surfaces and Spaces (ISS '16)*. ACM, 415–420. https://doi.org/10.1145/2992154.2996791

[3] Ahmed Sabbir Arif, Michel Pahud, Ken Hinckley, and Bill Buxton. 2014. Experimental Study of Stroke Shortcuts for a Touchscreen Keyboard with Gesture-Redundant Keys Removed. In *Proceedings of Graphics Interface 2014 (GI '14)*. Canadian Information Processing Society, Toronto, ON, Canada, 43–50.

[4] Ahmed Sabbir Arif and Wolfgang Stuerzlinger. 2009. Analysis of Text Entry Performance Metrics. In *2009 IEEE Toronto International Conference Science and Technology for Humanity (TIC-STH)*. 100–105. https://doi.org/10.1109/TIC-STH.2009.5444533

[5] Stuart K. Card, Thomas P. Moran, and Allen Newell. 1983. *The Psychology of Human-Computer Interaction*. CRC Press.

[6] Steven J. Castellucci, I. Scott MacKenzie, Mudit Misra, Laxmi Pandey, and Ahmed Sabbir Arif. 2019. TiltWriter: Design and Evaluation of a No-Touch Tilt-Based Text Entry Method for Handheld Devices. In *Proceedings of the 18th International Conference on Mobile and Ubiquitous Multimedia (MUM '19)*. ACM, 1–8. https://doi.org/10.1145/3365610.3365629 Article 7.

[7] Mikael Goldstein, Robert Book, Gunilla Alsiö, and Silvia Tessa. 1999. Non-Keyboard Qwerty Touch Typing: A Portable Input Interface for the Mobile User. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems (CHI '99)*. ACM, Pittsburgh, Pennsylvania, USA, 32–39. https://doi.org/10.1145/302979.302984

[8] Nathan Green, Jan Kruger, Chirag Faldu, and Robert St. Amant. 2004. A Reduced Qwerty Keyboard for Mobile Text Entry. In *CHI '04 Extended Abstracts on Human Factors in Computing Systems (CHI EA '04)*. ACM, New York, NY, USA, 1429–1432. https://doi.org/10.1145/985921.986082

[9] Anna-Maria Gueorguieva, Gulnar Rakhmetulla, and Ahmed Sabbir Arif. 2018. Enabling Input on Tiny/Headless Systems Using Morse Code. https://www.asarif.com/pub/Gueorguieva_CCBM2018_Morse_Code.pdf

[10] R. L. Hershman and W. A. Hillix. 1965. Data Processing in Typing: Typing Rate as a Function of Kind of Material and Amount Exposed. *Human Factors* 7, 5 (Oct. 1965), 483–492. https://doi.org/10.1177/001872086500700507

[11] Tomonari Kamba, Shawn A. Elson, Terry Harpold, Tim Stamper, and Piyawadee Sukaviriya. 1996. Using Small Screen Space More Efficiently. In *Proceedings of the SIGCHI conference on Human factors in computing systems common ground - CHI '96*. ACM Press, Vancouver, British Columbia, Canada, 383–390. https://doi.org/10.1145/238386.238582

[12] Per Ola Kristensson and Keith Vertanen. 2014. The Inviscid Text Entry Rate and Its Application as a Grand Goal for Mobile Text Entry. In *Proceedings of the 16th international conference on Human-computer interaction with mobile devices & services (MobileHCI '14)*. ACM, Toronto, ON, Canada, 335–338. https://doi.org/10.1145/2628363.2628405

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

[14] Frank Chun Yat Li, Richard T. Guy, Koji Yatani, and Khai N. Truong. 2011. The 1Line Keyboard: A QWERTY Layout in a Single Line. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST '11)*. ACM, New York, NY, USA, 461–470. https://doi.org/10.1145/2047196.2047257

[15] Yunzhi Li, Yinan Xu, Xiang Li, Jie Liu, and Nianlong Li. 2018. TapFlick: Combining Tap and Flick for Text Entry on Touchscreen Devices. In *Proceedings of the Sixth International Symposium of Chinese CHI (ChineseCHI '18)*. ACM, New York, NY, USA, 120–123. https://doi.org/10.1145/3202667.3202686 event-place: Montreal, QC, Canada.

[16] 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 (CHI EA '03)*. ACM, New York, NY, USA, 754–755. https://doi.org/10.1145/765891.765971

[17] I. Scott Mackenzie, Shawn X. Zhang, and R. William Soukoreff. 1999. Text Entry Using Soft Keyboards. *Behaviour & Information Technology* 18, 4 (Jan. 1999), 235–244. https://doi.org/10.1080/014492999118995 Publisher: Taylor & Francis.

[18] Gulnar Rakhmetulla and Ahmed Sabbir Arif. 2020. Senorita: A Chorded Keyboard for Sighted, Low Vision, and Blind Mobile Users. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI '20)*. ACM, Honolulu, HI, USA, 1–13. https://doi.org/10.1145/3313831.3376576

[19] Marco Romano, Luca Paolino, Genoveffa Tortora, and Giuliana Vitiello. 2014. The Tap and Slide Keyboard: A New Interaction Method for Mobile Device Text Entry. *International Journal of Human–Computer Interaction* 30, 12 (Dec. 2014), 935–945. https://doi.org/10.1080/10447318.2014.924349

[20] Yuan-Fu Shao, Masatoshi Chang-Ogimoto, Reinhard Pointner, Yu-Chih Lin, Chen-Ting Wu, and Mike Chen. 2016. SwipeKey: A Swipe-based Keyboard Design for Smartwatches. In *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '16)*. ACM, New York, NY, USA, 60–71. https://doi.org/10.1145/2935334.2935336

[21] Ken C.F. Tan, Edwin Ng, and Julian J.S. Oh. 2003. Development of a Qwerty-Type Reduced Keyboard System for Mobile Computing: Tengo. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 47, 6 (Oct. 2003), 855–859. https://doi.org/10.1177/154193120304700603

[22] Stephen P. Witte and Roger D. Cherry. 1986. Writing Processes and Written Products in Composition Research. In *Studying Writing: Linguistic Approaches* (1 ed.). Sage Publications, Beverly Hills, CA, USA, 112–153.