IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Efficient Typing on Ultrasmall Touch Screens with In Situ Decoder and Visual Feedback

KU BONG MIN[1], JINWOOK SEO[2](Senior Member, IEEE)
[1]Seoul National University, Department of Computer Science & Engineering, 1 Gwanak-ro Gwanak-gu (e-mail: kubong.min@hcil.snu.ac.kr)
[2]Seoul National University, Department of Computer Science & Engineering, 1 Gwanak-ro Gwanak-gu (e-mail: jseo@snu.ac.kr)

Corresponding author: Jinwook Seo (e-mail: jseo@snu.ac.kr).

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2019R1A2C2089062).

ABSTRACT Typing on a smartwatch is challenging because of the fat-finger problem. Rising to the challenge, we present a soft keyboard for ultrasmall touch screen devices with efficient visual feedback integrated with autocorrection and prediction techniques. After exploring the design space to support efficient typing on smartwatches, we designed a novel and space-saving text entry interface based on an in situ decoder and prediction function that can run in real-time on a smartwatch such as LG Watch Style. We outlined the details implemented through performance optimization techniques and released interface code, APIs, and libraries as open source. We examined the design decisions with the simulations and studied the visual feedback methods in terms of performance and user preferences. The experiment showed that users could type more accurately and quickly on the target device with our best-performing visual feedback design and implementation. The simulation result showed that the single word suggestion could yield a sufficiently high hit ratio using the optimized word suggestion algorithm.

INDEX TERMS Consumer electronics, human computer interaction, natural language processing, system analysis and design.

I. INTRODUCTION

WHEN using a smartwatch, it is often necessary to enter text, such as a quick reply to a text message or adding a reminder or an event. Commercial smartwatches support various methods such as voice recognition, predefined sentences for replying to a text message, and gestures for sharing messages with others. Even though these methods are useful in some cases, they also have pros and cons. For example, voice typing is not comfortable in public areas for some people. Commercial platforms began to support soft keyboards to overcome the limitations(e.g., Google Keyboard-Gboard on Android Wear 2.0 [12]).

In the HCI field, several studies have been conducted on text entry with a smartwatch. To accurately select a target key on a small touchscreen, various interaction techniques have been proposed that can be classified as multistep or indirect selection techniques [1], [3]–[6] while preserving the familiar QWERTY layout. Another way to overcome the fat-finger problem [13] when typing is to use a statistical decoder to select target keys probabilistically [8], [9], [11], [12]. Statistical decoder-based text entry opened up opportunities, demonstrating the surprisingly fast text entry speed with acceptable error rates on tiny touchscreens [8], [9], [11]. Most previous research has focused on improving the accuracy of decoders to see the performance improvement on soft keyboards of different sizes.

Besides designing a good statistical decoder, it is essential to design effective interaction techniques working with the decoder. Vertanen et al. showed that typing performance can vary with the design choices for possible interactions on the decoder-based smartwatch soft keyboard [11]. However, little attention has been paid to the design and evaluation of visual feedback techniques, such as displaying input characters for statistical decoder-based text entry on ultrasmall touchscreen devices. To our best knowledge, there has been no empirical evaluation on the effectiveness of visual feedback techniques for displaying typed input characters on soft keyboards with a statistical decoder. For the word suggestion design, little is known about the evaluation of word prediction when using N-gram language models, whereas the feature is commonly

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3081173, IEEE Access

TABLE 1: A summary of typing performance on ultrasmall devices. Error rates are collected based on character error rate (CER), and if total error rate (TER) is reported, it is addressed separately in parentheses.

Name	WPM	Error Rate (%)	Phrase Set	Layout	Size (mm^2)
ZoomBoard [1]	7.6–9.3	0.1	MacKenzie [2]	Qwerty	17 × 6
SwipeBoard [3]	9.1–19.6	— (TER = 13.3)	4-letter words	Qwerty	12 × 12
SplitBoard [4]	14.8	0.5 (TER = 7.5)	MacKenzie	Qwerty	16 × 6
Zshift [5]	9.1	0.9	MacKenzie	Qwerty	32 (wide)
DriftBoard [6]	8.77	1.13	MacKenzie	Qwerty	28 × 14
COMPASS [7]	9.3–12.5	0.0	MacKenzie	Circle	31 (diameter)
WatchWriter [8]	22.0	1.5	MacKenzie	Qwerty	33 (wide)
VelociTap [9]	34.9	10.7	Enron [10]	Qwerty	25 × 16
VelociWatch [11]	17.3	3.0	Twitter IV+OOV [11]	Qwerty	29 × 29
TwoSlot [11]	20.6	3.0	Twitter IV+OOV	Qwerty	29 × 29
VHA SHADE (ours)	25.3	0.9	Enron	Qwerty	30 (wide)

used on mobile device.

When it comes to devices with an ultrasmall screen, the input channel is very noisy due to the fat finger problem. A robust statistical autocorrection mechanism can compensate for the uncertainty of the input channel. With a robust autocorrection mechanism, the wrong character input can be corrected without using the back key when users continue to enter the following characters correctly. In this regard, we design a visual feedback method for character prediction to help the user understand the decoder's corrective behavior.

In this paper, we explore the previous research in the design space of a soft keyboard on an ultrasmall touchscreen device, and we present a novel QWERTY-based soft keyboard named Visual Hints for Accurate typing (VHA) with a state-of-the-art statistical decoder implementation and visual feedback for the decoded output shown in Fig. 1. Through an iterative design process, with the user study and the simulations, we select the most effective visual feedback and interaction for typing. In the user study, we evaluate potential visual feedback methods for a typed character. The result shows that users can type most precisely and quickly with the color-coded, decoded-output feedback (SHADE) shown in Fig. 3b. The character error rate (CER) is very low at 0.9% while preserving the fast typing speed of 25.3 wpm on the 30 mm wide smartwatch as shown in Table 1.

The main contribution of this paper consists of the following. First, we examine the design space of text entry for smartwatches and propose a space-efficient virtual keyboard layout design with the essential interactions. Second, we show how a powerful statistical decoder and predictor can run in real-time on embedded devices. We use novel approaches such as context-aware error model and augmented trie (prefix tree) data structure to optimize accuracy and runtime performance. Third, we design and implement an API that can easily use the statistical decoder and predictor. Until now, user interaction studies about the statistical decoder and predictor have been difficult because not enough code has been publically available. We share our implementation, including the API and libraries, on GitHub [14] to help other researchers investigate design spaces and interactions. Fourth, we design and implement prediction functions optimized for smartwatches based on the observation that single-word prediction based on an N-gram language model can compensate for hit ratio and keystroke reduction compared to multi-word prediction based on a unigram language model. Finally, we design visual feedback methods for the decoded output with character- and word-level decoders for soft keyboards. We compared four feedback methods in a controlled user study, and the result shows that our proposed method-SHADE enables users to type more accurately and quickly without preference degradation.

II. RELATED WORK

We reviewed previous studies and classified them systematically to make a design space for the soft keyboard on the smartwatch. We explore the design space along two dimensions. One is the operation to support, and the other is the execution level. For the operations, we categorize them into visual feedback method for the typed text, suggestion method, and typo recovery. For the execution level, there are possible three levels—character, word, and sentence. Table 2 shows the design space for typing interactions on ultrasmall devices, mostly using a touchscreen-based smartwatch. While exploring the design space, we also formulate our research questions in the context of the previous related work.

A. INTERACTIONS AND VISUAL FEEDBACKS FOR TYPING

For the visual feedback of typing, two approaches are possible. One is to display an accurately selected target, and the other is to show a statistically decoded output considering touch inaccuracy because of the small touchscreen size.

To support accurate tapping on the target key on a small screen, several approaches have been proposed in previous studies. First, multistep selection approaches are available. ZoomBoard [1] and SplitBoard [4] are based on zooming and tapping approaches. SwipeBoard [3] uses two successive swipe gestures to select a target key in the QWERTY keyboard. Second, indirect selection can be another option. Zshift [5] uses a callout-based shift-pointing technique, and DriftBoard [6] uses a panning-based technique that utilizes a fixed cursor point with a movable QWERTY layout. Back-of-device [19] and tilt-based gesture [20] can be other options

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3081173, IEEE Access

TABLE 2: Design space for typing interaction on ultrasmall devices.

Operations / Execution level	Visual feedback of typing: with selected target	Visual feedback of typing: with decoded output	Suggestions	Typo recovery
Characters	**Multi-step selection** *ZoomBoard* (Oney et al, 2013) [1] *SwipeBoard* (Chen et al., 2014) [3] *SplitBoard* (Hong et al., 2015) [4] **Indirect selection** *Zshift* (Leiva et al., 2015) [5] *DriftBoard* (Shibata et al., 2016) [6] **Ambiguous selection** *COMPASS* (Yi et al., 2017) [7]	**Touch (Offset) modeling** *WatchWriter* (Gordon et al., 2016) [8] *Gboard for Android Wear [12]* **Noisy channel model** *VelociTap* (Vertanen et al., 2015) [9] *VelociWatch* (Vertanen et al., 2019) [11] *TwoSlot* (Vertanen et al., 2019) [11] *VHA* (ours)	**Based on user's conceptual model of decoder** *ForceType* (Weir et al., 2014) [15] **For visual hint on likelihood** *COMPASS* *VHA*	**Deletion of the last character** *ZoomBoard* *SwipeBoard* *DriftBoard* **Undo latest decoding** *VelociTap* *VelociWatch*
Words		**Noisy channel model** *VelociTap* *VelociWatch* *TwoSlot* *Gboard for Android Wear* *VHA*	**For correction & prediction** *WatchWriter* *Gboard for Android Wear* *VelociWatch* *TwoSlot* *VHA*	**Editing on character-level.** *Gboard for Android Wear* **Editng on word-level.** *WatchWriter* *TwoSlot* *VHA*
Sentences	**Predefined Sentences** *Microsoft Band* [16] *Apple Watch* [17] *Android Wear* [18]	**Noisy channel model** *VelociTap* *VelociWatch* *TwoSlot* *Microsoft Band Keyboard* *VHA*	**For selecting words** *Microsoft Band Keyboard* **For predicted replies on** *Apple Watch* *Android Wear*	

if additional input channels are available. Third, it is also possible to allow an ambiguous selection by assigning multiple characters on a single key. COMPASS [7] uses multiple cursors on a circular keyboard to select keys by rotating the bezel. DualKey [21] allocates two characters in a key and determines the target key by a finger-identification method. Most commercial platforms, such as Microsoft Band [16], Apple Watch [17], and Google Android Wear [18], support predefined sentence selection for messaging applications in addition to typing in character units. For example, Apple watch supports user-defined messages in addition to default messages such as "What's up?", "I'm on my way.".

The statistical decoder-based text-input method [22] has strong potential because it has shown a much faster input speed than multi- or indirect-selection-based techniques on smartwatch-sized screens, as shown in Table 1. VelociTap [9] reported 34.9 wpm on a 25 mm wide smartwatch-sized keyboard at a 10.7 % character error rate (CER), and WatchWriter [8] reported 22.0 wpm with a CER close to 1.5%.

At the character input level, text entries can use touch models to select characters for display. For example, Weir et al. [15] used the Gaussian process regression approach [23] to make a personalized touch model, and Yi et al. [24] modeled touch offset and its variation with bivariate Gaussian distribution [22], [25] on a smartwatch. Based on these touch models, decoder-based text entries, such as WatchWriter and Gboard for Android Wear, can display characters on the screen as visual feedback for each touch input. In addition to this touch model, VelociTap has constructed Noisy Channel Model [26], [27] that uses both a touch model and a character language model to decode touch input to the displayed characters.

When the touch input is very uncertain, displaying selected characters on the screen can frustrate users, as shown in Fig. 3d. As far as we know, there have been few comparative user studies to evaluate different visual feedback methods for selected and decoded characters on ultrasmall screen devices. So our first research question (**Q1**) is as follows. **What is the most effective visual feedback method for displaying typed characters on an ultrasmall soft keyboard in terms of typing speed, accuracy, and user preferences?**

At the word input level, there are two types of displaying decoded words. Most decoder-aided soft keyboards on mobile devices, including WatchWriter and Gboard for Android Wear, display the decoded word in the suggestion bar. VelociTap and VelociWatch directly display the decoded word in the textbox using autocorrection.

At the sentence input level, VelociTap supports sentence-level decoding to modify words in a sentence when the user finishes typing them. The Microsoft Band keyboard displays sentence-level input feedback on a separated full-screen mode that maximizes the size of the keyboard layout on the screen.

B. INTERACTIONS FOR SUGGESTIONS

Interactions for suggestions are familiar to most users. For example, many mobile text entries support word suggestions using a suggestion bar. In some situations, character-level interactions are possible. ForceType [15] designs force-based interactions based on the user's conceptual model about the decoder to adjust the autocorrection threshold dynamically. COMPASS uses a visual hint that contains the likelihood of

each character; It allocates a visually salient color on the character keys according to their probability. As a result, it guides users to focus on the possible characters in the dictionary.

There are two types of word-level suggestion methods: correction based on a decoder and prediction (i.e., completion) using a language model [8]. Interactions for sentence-level suggestions are also possible. The Microsoft Band keyboard allows users to modify decoded sentences by touching their content. Commercial wearable platforms, including Microsoft Band, Apple Watch, and Google Android Wear, also support suggesting predefined sentences for quick replies [16]–[18].

Text prediction is one of the most widely used techniques to enhance the communication rate in mobile computing as well as in augmentative and alternative communication (AAC) [28]. Quinn and Zhai [29] studied the effect of word prediction in a mobile touchscreen device with a separate suggestion bar. Although the number of keyboard actions for text input decreased and the suggestion bar was subjectively favored, it was reported that the average time for text input was degraded because of additional costs for the users' attention to the suggestion bar and decision making. However, the N-gram language model could improve the accuracy of word prediction, which has a positive correlation with the average hit ratio of word prediction. So our second research question (**Q2**) is as follows. **How much can the N-gram language model raise the hit ratio of word prediction? Is it enough to replace the multi-words selection with a single-word one?**

C. INTERACTIONS ON TYPO RECOVERY

A back key is necessary because typing errors are inevitable. However, due to the high cost of adjusting the typing error, fixing a typo is an undesirable action for users [30]. It has been a good research topic to lessen the cost. For example, there was a study to facilitate the correction of overlooked errors using the smart restorable backspace technique [31]. When users deleted remaining words to correct a typo in the middle of a sentence, the technique helps users correct overlooked typos by suggesting the remaining words not to retype them.

When it comes to statistical decoder-driven text entry, the role of the back key is more than just deleting the target element because it needs to restore the previous status to update the decoded output along with the word suggestion list into the last stage. Moreover, allowing cursor offset movement in any location offers a greater number of design choices to consider, for example, whether word correction and prediction are turned on or off.

In this regard, ZoomBoard, SwipeBoard, and DriftBoard, which do not depend on a decoder, use a back key for deleting the last input character. In contrast, VelociTap and VelociWatch use a back key to undo the previous character input. Gboard for Android Wear has more flexible operations that allow not only undoing the last input character but also moving cursors without turning off word correction and prediction. Meanwhile, WatchWriter and TwoSlot, a simpler version of VelociWatch, intentionally prevents character-by-character operations to enable word-level decoding. Thus, the back key in WatchWriter and TwoSlot erase words on a word-by-word basis.

In the case of typing on the soft keyboard using the statistical decoder, users can correct typos in two ways. The first way is to use the back key to clear and retype the word as soon as possible. The second way is to keep typing, even if users find a typo, and let the soft keyboard correct it using the autocorrection function.

III. DESIGN

As shown in Fig. 1, we designed and implemented the VHA soft keyboard on the Android platform to find answers to our research questions.

A. LAYOUT AND INTERACTION DESIGN

Fig. 2 shows the layout and interaction design with gaze points highlighted. We used the standard QWERTY layout, which is familiar to most mobile-device users. Because screen assets are limited in ultrasmall screen devices, design decisions focus on minimizing the screen space required for the QWERTY layout and control keys, ensuring that the remaining screen area for information display is as large as possible. On top of the QWERTY keyboard, an input textbox is located showing the predicted word as well as decoded characters. Above it, the presentation textbox, which has a gray background, is located to show the presented sentence for the user experiment. On the right side of these two textboxes, we located a button to indicate the end of the sentence, similar to the enter key.

Whereas the presentation box is used for the user experiment to present a given sentence in the implementation, the valuable space can be used for other purposes also. For example, we can suggest a predefined message based on context or present useful information to aid the user's task.

During the VHA soft keyboard development, we observed that the space bar and the back key could be accidentally selected due to the fat finger problem when they were assigned to touch keys. Similarly, Vertanen et al. also addressed that the spacebar at the bottom of the keyboard would cause accuracy problems due to size limits on their velociWatch design [11]. This leads to user frustration and a noticeable performance degradation because, in the decoder-based soft keyboard, the role of the space bar or back key is not only to add spaces or delete characters but also to perform or undo decoding, respectively. Therefore, we decided to separate the input channels between the alphabet and control keys. We mapped control keys to different input channels—swipe gestures like ZoomBoard [1], VelociTap [9], Swype [32], and velociWatch [11]: (1) swipe-right for space bar, (2) swipe-left for back key, (3) swipe-down for word completion, and (4) swipe-up reserved for future usage (e.g., turn-off/on autocorrection).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3081173, IEEE Access

(a) (b) (c) (d) (e)

FIGURE 1: A storyboard of our VHA design when typing a sentence. (a) The user touched the *t* key on a QWERTY keyboard. The *h*, *o* keys are highlighted to indicate that they are the most expected next key candidates in the current input sequence. (b) The user touched the "omptto" keys, but the autocorrected characters, "omorro" are displayed. The corrected characters are shown in gray color, and the predicted word "tomorrow" is displayed with light-gray color on the "w." (c) After touching the *w* key, the user finished typing "tomorrow" with the right-swipe gesture. (d) The user touched the "kf" keys, and "of" was displayed with character correction. (e) After the swiping to the right, the corrected word "if" was shown (aided by a word corrector), and the user continued to type the next word. A predicted word, "possible," was shown.

(a) (b)

FIGURE 2: The virtual keyboard layout is shown in the circular smartwatch; (a) our soft keyboard (VHA) and (b) Google Keyboard (Gboard) on Android Wear 2.0. Possible gaze points to enter a word are depicted with a yellow eye icon. The VHA keyboard on the left shows additional information, such as a presented phrase for the user experiments in the gray textbox located at the top. On the right side, there is the end button, which is used to finish entering a phrase. The input textbox is located below the gray textbox showing the input words as well as a predicted word. The cursor is located as a blinking underline at the end of the actual input text. At the bottom of the soft keyboard, there is a QWERTY layout to enter characters. Bolded yellow keys are predicted character candidates for the next input. The blue arrow means swiping gestures. The left arrow means swipe-left gesture acting as a back key, and the right arrow means swipe-right gesture acting as a spacebar, and the down arrow means swipe-down gesture triggering a word completion. The Gboard on the right has no area to show information at the top. Instead, the input textbox is located at the top showing last input words, and at the end of the input textbox, there is a blinking cursor as a vertical bar. On the right side of the input textbox, there is a rounded button to finish entering a sentence. Below the input textbox, the suggestion bar shows suggested words for fixing or predictive typing. Users can see more suggested words by touching the downward arrow button. Below the input textbox, the QWERTY layout is to shown to enter characters on top of the special keys for mode change, spacebar, and back key.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3081173, IEEE Access

The space bar key has two functions. First, it acts as a word delimiter to trigger autocorrection with a word-level decoder, and second, it inserts the space character after the decoded word.

We initially decided to use the back key (swipe-left gesture) for an undo operation. When users swipe-left, the application discards the latest key and goes back to the previous state. For example, when the last input was a user touch on an alphabet key, it just discards the last character-level decoding output and rolls back to the sequence of characters that were previously returned by the character-level decoder. If the last input was a swipe-right gesture for entering a blank space, the system clears the inserted space; at the same time, it cancels the word-level decoding output and replaces it with the output from the underlying character-level decoder.

However, when users type a word and see an unintended sequence of characters entered, they have two options: to undo it with the swipe-left gesture or to continue typing the next character carefully when they think the system can modify the input correctly in the end. During the pilot study, we observed that some users tend to use the swipe-left gesture frequently even when the decoder can eventually correct the input sequence. Moreover, if an error occurs at the end of entering a word, rewriting the entire word is often more efficient than undoing a few characters. The previous study by Vertanen et al. also showed that the TwoSlot method, a simpler version of VelociWatch, which blocked the deletion of individual characters like WatchWriter, outperformed VelociWatch that enables character-level removal [11]. Therefore, we decided to get rid of character-decoder-level undo operations in our system. Instead, our system clears out all the decoded character sequences that came from the character-level decoder and resets the decoder to make a user enter the correct word again.

For word completion, users can select the suggested word with a swipe-down gesture when only one word is suggested. We also had tested direct touching gesture on the output textbox in place of the swipe-down gesture, but we observed that there was no significant improvement in typing speed and error rate. We decided to use the swipe-down gesture instead of touching the textbox, considering the consistency for the other control events.

B. VISUAL FEEDBACK FOR DECODED CHARACTERS

We explored four possible design choices for displaying input text, except for trivial cases such as displaying "*" for each key touch. Fig. 3 shows the four design choices. The first design choice is to display the decoded characters instead of the nearest keys based on the touch model (Fig. 3a). We expected that displaying decoded characters is helpful on high-ambiguity input channels to prevent unnecessary undo operations when the decoder keeps track of the user's intentions well. The second is to add color-coding to the first method to provide additional information about the decoded output. We assigned black if the decoded character is the same as the touched key. Other decoded characters (not equal to the touched key) are grayed out (Fig. 3b). We expected that color-coded information would help users build a more precise mental model of our decoder. The third is to use "*" for the decoded character that is not the same as the touched key instead of graying it out (Fig. 3c). With this design option, we wanted to investigate if the user could trust the decoder more with less information provided. We also wanted to know the relationship between users' cognitive burden and the level of details in the information displayed. The last option is to display the touched key with a touch model, like WatchWriter, Gboard for Android Wear, and most soft keyboards on mobile devices (Fig. 3d). Key selection can be made in several ways, for example, by touching within the visual boundary of a key or by choosing the nearest key to the point of the touch. We used the bivariate Gaussian distribution as a touch model, using the Bayesian Touch Distance metric to decide the intended key [25].

We ran a controlled user study experiment to make the best design decision on visual feedback for decoded characters, answering **Q1** in section V.

C. VISUAL CUE FOR CHARACTER PREDICTION

We designed a visual cue to highlight the probable next input characters on a QWERTY keyboard layout. The probable characters are selected based on the character-level prediction algorithm that is described in the following section IV-D. Although character-based prediction does not help in shortening keystrokes per character (KSPC), we found two benefits during the pilot study. First, it can help users type characters more confidently because the visual cues could facilitate users' recognizing how well the decoders predict the intended words. Second, it would be able to serve as a guide to find target keys for users who are not familiar with the small QWERTY layout. For this reason, we made the visual cue salient to users by using distinct size, boldness, and hue (see Fig. 2a). Yi et al. also reported users' preference for the visual hint for character selection on their rotational keyboard on non-touch smartwatches [7].

D. WORD PREDICTION

Because screen asset on ultrasmall screen device is very limited, cost-benefit analysis is needed to decide a proper number of suggested words. Quinn and Zhai reported that increasing the assertiveness of suggestions for text entry reduced KSPC and was preferred by users whereas it lowered the average entry rate. They defined the assertiveness as the tendency for an interface to present itself. With extraverted assertiveness, the interface suggests every possible word. With introverted assertiveness, the interface never suggests words. With ambiverted assertiveness, suggestion results are gated by a probability threshold to suggest only high probability words [29]. To suggest words, they used a prediction model based on the frequency information of words (i.e., unigram language model) in a dictionary. They discussed that N-gram word prediction and input error correction could save

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3081173, IEEE Access

(a) DECODED (b) SHADE (c) STAR (d) TOUCH

FIGURE 3: When a user tried to type "today", but actually typed "yofay", conditions that can display transcribed characters on the screen are shown as follows: (a) the decoded characters only (DECODED), (b) the decoded characters with color-coding to identify nearest characters from the touchpoints (SHADE), and (c) "*" for decoded characters that are not nearest to the touchpoints (STAR), (d) the nearest characters from the touchpoints using the Bayesian Touch Distance metric (TOUCH).

more keystrokes with similar costs of attending, deciding, and selecting the suggested words.

As the suggestion interface requires additional screen space and time to perform, we designed a new interface for word suggestion to minimize the space and time requirement while trying to preserve most benefits. Fig. 1e shows our interface for word suggestion. According to the simulation result on section VI-B, we decide to suggest only the most probable word at a time. When users start typing a word, the suggested word appears in the input text box if the language model reports a higher probability than the predefined threshold for the word. The suggested word is located just after the cursor. It is in gray to indicate that the user has not yet entered it. We used 4-gram word prediction to raise the hit ratio of the suggested words considering the performance enhancement against the increased model size. As a result, the interface does not require additional user interface components for displaying and selecting the suggested words. We implemented two interactions for users' selection of the suggested word—a direct tap on the output textbox and a swipe-down gesture. Then we compared the two in terms of typing speed and error rate, and there was no significant difference between them. Vertanen et al. also compared tap and swipe gestures on their VelociWatch implementation, and they also reported that there was no significant difference between tap and swipe. Therefore, we decided to use the latter as the selection interaction, considering that we already used swipe gestures for the back and space key.

IV. IMPLEMENTATION

In this section, we explain the implementation details of our VHA method. Most interface components are based on Java, whereas decoding and prediction were implemented in C++ with the Android native development kit [33] to meet the real-time constraints of the task.

A. LANGUAGE MODEL

To construct a language model, we collected 103 million (103,790,367) tweets with the Twitter streaming API [34] because Twitter is known as a good source for a large number of conversational phrases [35]. We filtered out retweets, and then classified the tweets into personal or nonpersonal tweets by using a Naïve Bayesian classifier. We manually labeled 400 tweets to train the classifier. The classifier, which showed 92% accuracy, used both content features and context features. As the content features, we used the top 100 most common words, number of URLs, number of hashtags, number of users mentioned, number of symbols. As the context features, we used the number of followers, the number of friends, favorited, geo-tagged, and timezone. Finally, automatically transferred URLs that start with "http://t.co" were deleted from the text. Exactly 36,657,728 tweets contributed to building our domain corpus with 500,090,467 words.

Based on the corpus, we generated a word-level 4-gram model and a character-level 7-gram model using the SRI Language Modeling (SRILM) toolkit [36]. For the word model, Moby's word list [37] was used as a vocabulary. Witten-Bell smoothing [38], which is known to be more robust than modified Kneser-Ney [39] for entropy-based downsizing [40], was used to build up the models. After downsizing the models using entropy pruning provided by SRILM, we used KenLM [41] on the target device to query language models in real-time.

B. ERROR MODEL

To make a high-quality word decoder (corrector), we used both misspelling and touch models. We gathered misspelling data, which contains both cognitive and typing errors from the following four sources: (1) Wikipedia's lists of common misspellings (3,246 words; 4,497 misspellings) [42], (2) Roger's Misspellings (7,841 words; 39,709 misspellings) [43], (3) Aspell Misspellings (466 words, 547 misspellings) [44], and (4) MSR misspellings (10,521 words; 31,180 misspellings) [45].

Based on the collected data, we built a context-aware confusion matrix with modified Kneser-Ney smoothing [39]. We used the previous two characters before a typo as a context. We also calculated the distribution of error locations

within the words from the collected data, as in the Baba and Suzuki's study [45]. We normalized the CER to an average of 2% [46].

For touch modeling, we used a dual Gaussian distribution model that can reliably predict touch accuracy along with target size [47].

C. DECODER

Algorithm 1: Decoding based on the noisy channel model

Require: context C of n elements; input V; return size K

1: $results \leftarrow []$
2: $candidates \leftarrow GenerateDecodeCandidates(V)$
3: **for all** W in $candidates$ **do**
4: $\quad LMProbability \leftarrow QueryLanguageModel(C, W)$
5: $\quad EMProbability \leftarrow QueryErrorModel(C, V, W)$
6: $\quad$ append $(W, EMProbability * LMProbability^{\lambda})$ to $results$
7: **end for**
8: sort $results$ by probability in decreasing order
9: **return** $results[:K]$

We used the noisy channel model to make decoders [26], [27], which are used for character-, word-, and sentence-level correction for our VHA implementation. Using the Bay's theorem, we can estimate the intended correction w, from a typo v by finding the correction w that maximizes $Pr(w) \times Pr(v|w)$. We calculated the prior $Pr(w)$ and the channel probability (conditional probability) $Pr(v|w)$ from the language model and the error model, respectively. As Brill and Moore reported that channel and language models directly affect the quality of the decoder [48], we constructed the language and error models elaborately as described in section IV-A and IV-B.

Algorithm 1 shows the decoding procedure. The character decoder generated all substitution (including itself), insertion, and deletion cases, regardless of the given input V (line 2). For each case, it calculated source probability (line 4) and channel probability (line 5) with the language model and error model, respectively. We weighted the source probability with λ as 1.2, which is empirically determined from our simulation (line 6). To determine the parameter λ, we implemented the word corrector along to this decoder algorithm based on the language and error models that are explained in the previous sections. We used selected sentences from Twitter and generated typos based on the error model. With these generated input sentences, we simulated word correction, and the best-performed value was selected.

Generating candidate words for word decoding is also important to improve the decoder quality. Because of the high uncertainty of the input channel, i.e., finger touch on the small screen, many typos exceeded the edit distance of two. However, even with the edit distance of two, if the length of a word is long, the number of candidates can be too large to process in real-time. Thus, we used beam pruning to control the number of candidates; for each character input, we maintained a string list that contains up to 50 candidates made by the character decoder with the previous list. When the user starts decoding a word using a swipe-right gesture, the word-level decoder will generate more candidate words from the strings that are generated by the character-level decoder in the candidate list: (1) every word in the vocabulary that has edit distance of one from the strings in the list, (2) every word in the vocabulary that has the same phonetic index as the input sequence using Double Metaphone [49]. The candidate words generated through this process can cover more than two edit distance, and thus it can help us produce proper decoding results without compromising time and accuracy.

D. PREDICTION

Algorithm 2: Prediction based on language model

Require: context C of n elements; prefix P; return size K

1: $results \leftarrow []$
2: $candidates \leftarrow GeneratePredictCandidates(P)$
3: **for all** W in $candidates$ **do**
4: $\quad probability \leftarrow QueryLanguageModel(C, P)$
5: $\quad$ append $(W, probability)$ to $results$
6: **end for**
7: sort $results$ by probability in decreasing order
8: **return** $results[:K]$

The prediction procedure is described in Algorithm 2. For character prediction, prediction candidates (line 2) are generated just by listing the entire alphabet. Given that the number of alphabets (26) is small, there is no performance problem in forecasting. However, the number of candidates for word prediction is the size of the whole vocabulary in the worst case. To make the prediction run in real-time, we constructed a modified trie (prefix tree) data structure from the dictionary. Each node additionally keeps the highest rank of its subtree to prune the tree by order of words easily. This allows us to limit the maximum number of words by the rank of the word frequency on the vocabulary to consider at runtime. Based on the simulation result in section VI-A, we selected the number of candidate words as 1,000, which achieves useful keystrokes saving ratio while not harming the real-time constraint.

V. EXPERIMENT1: USER STUDY FOR VISUAL FEEDBACK OF DECODING

There are a few possible design alternatives to display characters in the input textbox when users type a sentence into the smartwatch. In order to answer the research question **Q1**, that is, to make a design decision about the visual feedback of input characters, we conducted a user study to compare the performance, perceived workload, and user preference of these designs.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3081173, IEEE Access

FIGURE 4: The participants wore the smartwatch on their wrists to use the finger of their dominant hand.

A. APPARATUS AND PARTICIPANTS

The first study was performed on the 3.05 cm (1.2 in) LG Watch Style (LG-W270) [50]. The device supports a circular display with a resolution of 320 × 320. We used our VHA soft keyboard application running on the Android Wear OS 2.0 with Snapdragon Wear 2100 [51] AP (application processor) and 512 MB of RAM.

We recruited 12 participants (6 male, 6 female) on our campus. Their ages ranged from 23 to 33 years (mean = 25.8, sd = 3.1). They have used smartphones for over a year and are accustomed to reading and typing in English. Each participant was rewarded with approximately US $10.

B. PROCEDURE

After explaining the goals of the experiment, we asked the participants to sign the consent form. Tasks were demonstrated with explanations before we handed them the smartwatch. The participants were seated and wore the smartwatch on their wrists to use the finger of their dominant hand (see Fig. 4). They practiced all conditions in each training session with Q&A. After these training sessions, the participants completed the conditions in the counterbalanced order.

In all conditions, the participants were asked to enter the given phrases as quickly and precisely as possible. They were able to relax between phrases. The participants were shown 10 random memorable phrases that only contain alphabets from Enron Mobile phrases. Enron Mobile phrases collected from the genuine mobile e-mails to evaluate the performance in casual text entry tasks (e.g., messaging with a smartwatch) [10]. The test application recorded visual feedback conditions, presented sentences, prescribed sentences, and touch locations with timing information to calculate entry and error rates. After finishing each condition, a NASA-TLX [52] and subjective rating using a five-point Likert scale questionnaire were administered. The participants were allowed to have break time between conditions as long as they want. On average, the study took about 50 minutes.

There were four conditions for the user study as presented in Fig. 3: (a) displaying decoded characters (DECODED), (b) displaying decoded characters that are not nearest to the touchpoints in a distinctive color (SHADE), (c) displaying "*" for decoded characters that are not nearest to the touchpoints (STAR), and (d) displaying the nearest characters from the touchpoints (TOUCH). We ran the study as a within-subject design where each participant performed tasks under all conditions. There was a single independent variable with the four levels.

TABLE 3: Statistics for participant's entry rate and error rate.

	Entry Rate*		Error Rate (%)*	
Technique	**M (SE)**	**95% CI**	**M (SE)**	**95% CI**
DECODED	23.6 (1.7)	[19.8 27.4]	2.4 (.7)	[1.0, 3.8]
SHADE	**25.3** (1.7)	[21.6, 29.0]	.9 (.3)	[.4, 1.5]
STAR	24.0 (1.3)	[21.1, 26.8]	1.0 (.3)	[.2, 1.8]
TOUCH	**21.9** (1.5)	[18.6, 25.0]	1.4 (.4)	[.8, 2.0]

C. RESULTS

In this section, we describe the result of the first experiment comparing the performance, perceived workload, and user preference of the four visual feedback conditions. We gathered 480 phrases in total (12 participants × 4 conditions × 10 phrases).

1) Entry rate

We measured words per minute (wpm). A word is defined as five characters, including spaces, as in MacKenzie's work [53]. We measured the time between the first touch and the last event, just before touching the end button. The grand mean of entry rate was 23.7 wpm; the fastest was SHADE at 25.3 wpm, STAR at 24.0 wpm, DECODED at 23.6 wpm, and the slowest was TOUCH at 21.8 wpm (Fig. 5 and Table 3). Using repeated measures ANOVA, the main effect of the visual feedback method on entry rate was statistically significant ($F_{3,33} = 3.32$, $p < .05$, $\eta_p^2 = .23$). Post hoc comparisons using the Bonferroni correction conducted. There was a statistically significant difference between SHADE and TOUCH ($M = 3.5$, $p < .05$, 95% CI [.0, 7.0]). There were no statistically significant differences for all the other pairwise comparisons.

2) Error rate

We calculated the error rate using CER [54], which is based on the minimum string distance method. The grand mean of error rate was 1.4%; the lowest was SHADE at 0.9%, STAR at 1.0%, TOUCH at 1.4%, and the highest was DECODED at 2.4% (Fig. 5 and Table 3). Using repeated measure ANOVA, the main effect of the visual feedback method on error rate was statistically significant ($F_{3,33} = 3.82$, $p < .05$, $\eta_p^2 = .26$). Post hoc comparisons using Bonferroni correction revealed that there were no statistically significant differences for all the pairwise comparisons.

3) Subjective ratings

We collected participants' subjective ratings on perceived preference, accuracy, and speed using a five-point Likert scale. Table 4 shows the average ratings and statistics for each

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3081173, IEEE Access

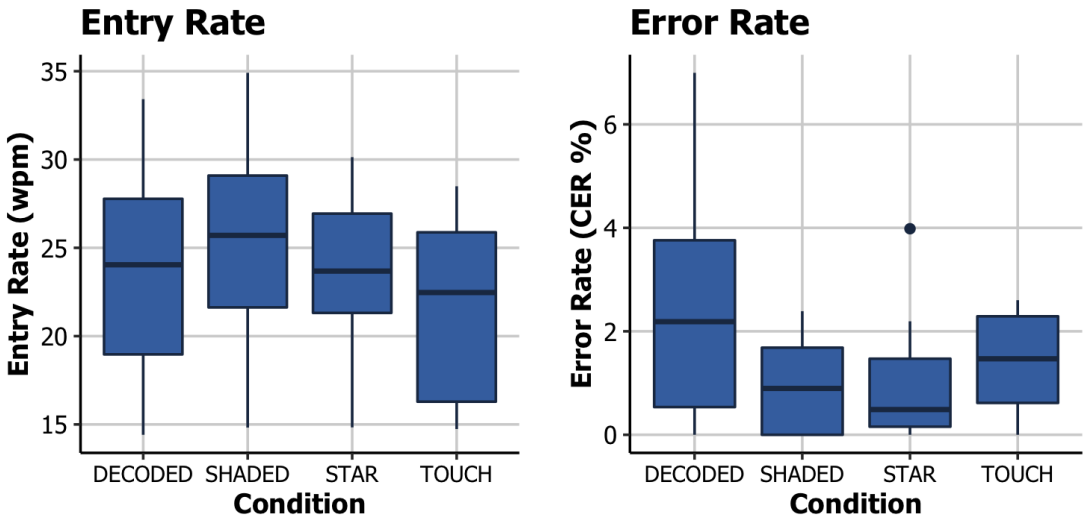


FIGURE 5: Participants' entry rate (left) and error rate (right).

TABLE 4: Average subjective ratings (1–most negative, 5–most positive) and statistics from Freidman tests.

Technique	DECODED	SHADE	STAR	TOUCH	χ^2	p
Preference*	**3.8**	**3.7**	**2.7**	3.3	10.47	.015
Accuracy	3.4	3.5	2.9	3.3	2.84	.416
Speed	3.7	3.8	3.3	3.4	3.49	.322

TABLE 5: Average NASA-TLX score (lower is better.) and 95% CI for different conditions.

Technique	DECODED	SHADE	STAR	TOUCH
Mental*	7.4 [2.6, 12.2]	7.8 [4.0, 11.6]	12.1 [6.4, 17.9]	11.9 [7.7, 16.1]
Physical	3.1 [-.6, 6.2]	3.9 [.8, 6.9]	4.7 [.7, 8.7]	4.3 [1.5, 7.1]
Temporal	7.9 [2.5, 13.4]	6.8 [1.8, 11.7]	6.4 [4.0, 8.8]	11.1 [4.7, 17.6]
Performance	7.2 [3.9, 10.5]	7.1 [3.9, 10.3]	6.9 [4.4, 9.5]	6.2 [3.3, 9.2]
Effort*	9.1 [3.0, 15.1]	5.1 [1.2, 9.0]	11.4 [5.3, 17.6]	12.1 [6.4, 17.8]
Frustration	6.5 [.9, 12.0]	6.3 [.8, 11.9]	12.56 [6.5, 18.6]	8.5 [1.8, 15.1]
Total*	41.2 [24.4, 57.9]	**37.0** [23.9, 50.1]	54.2 [39.9, 68.48]	**54.1** [40.7, 67.4]

question. The perceived preference levels for DECODED, SHADE, STAR, and TOUCH were 3.8, 3.7, 2.7, and 3.3, respectively. Using Freidman test, there was a statistically significant difference in perceived preference depending on visual feedback conditions ($\chi^2 = 10.47$, $p < .05$, $df = 3$). Post hoc analysis with Wilcoxon signed-rank tests was conducted. There were statistically significant different ratings between STAR and SHADE ($Z = -2.36$, $p < .05$) and between STAR and DECODED ($Z = -2.55$, $p < .05$). For all the other pairwise comparisons, there were no statistically significant differences in rating. The perceived accuracy levels for DECODED, SHADE, STAR, and TOUCH were 3.4, 3.5, 2.9, and 3.3, respectively. Using Freidman test, there was no statistically significant difference in perceived accuracy levels depending on visual feedback conditions ($\chi^2 = 2.84$, ns, $df = 3$). The perceived speed levels for DECODED, SHADE, STAR, and TOUCH were 3.7, 3.8, 3.3, and 3.4, respectively. Using Freidman test, there was no statistically significant difference in perceived speed levels depending on visual feedback conditions ($\chi^2 = 3.49$, ns, $df = 3$).

For NASA-TLX surveys, Table 5 shows the participants' scores and 95% CI. The total scores for DECODED, SHADE, STAR, and TOUCH were 41.2, 37.0, 54.2, and 54.1, respectively. Using repeated measure ANOVA, the main effect of the visual feedback method was statistically significant ($F_{3,33} = 5.64$, $p < .01$, $\eta_p^2 = .34$). Post hoc comparisons using Bonferroni correction revealed that there was a statistically significant difference between SHADE and TOUCH ($M = -17.1$, $p < .01$, 95% CI [–29.4, –4.8]). For all the other pairwise comparisons, there were no statistically significant differences in total scores. For the mental scores, the main effect of the visual feedback method was statistically significant ($F_{3,33} = 3.38$, $p < .05$, $\eta_p^2 = .24$) using repeated measure ANOVA. Post hoc comparisons using Bonferroni correction revealed no statistically significant differences for all the pairwise comparisons. For the effort scores, the main effect of the visual feedback method was statistically significant ($F_{3,33} = 3.45$, $p < .05$, $\eta_p^2 = .24$) using repeated measure ANOVA. Post hoc comparisons using Bonferroni correction revealed no statistically significant differences for all the pairwise comparisons. For the physical, temporal, performance, and frustrations scores, there were no statistically significant differences for the main effect of the visual feedback method.

D. DISCUSSION

The answer to **Q1** is, "**The SHADE method is the most effective visual feedback method among possible design options for displaying typed characters on an ultrasmall soft keyboard in terms of typing speed, accuracy, and user preferences.**"

It is noteworthy that the differences in the visual feedback methods alone make significant differences in typing speed and accuracy as well as subjective ratings. For the entry rate, SHADE showed the best performance among the four conditions (Table 3); and SHADE was significantly faster

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3081173, IEEE Access

than TOUCH. In the case of TOUCH, when users have chosen a key that was not intended by them, the output textbox immediately shows the wrong character even the decoder can autocorrect it. As a result, users might have to slow down their input rate to select keys more carefully or re-enter it after erasing the input.

For the error rate, SHADE also showed the lowest error rate among the four conditions, whereas DECODED showed the highest error rate (Table 3). In the case of DECODED, when a user selects an incorrect key, it is possible that the decoder autocorrects it into an intended key. In general, the decoder can correct commonly used phrases well, but uncommon phrases need to be entered more accurately. In the case of DECODED, the output textbox shows the decoded character, and the user cannot distinguish whether the displayed character is correctly selected or not. As a result, this can make an increased error rate because of the imprecise feedback for their touch input. We suspect that accurate visual feedback on a user's touch can help them select a key more precisely in case of entering uncommon phrases.

For the subjective ratings, there were no statistically significant differences for perceived speed and accuracy, whereas there was a statistically significant difference for perceived preference. STAR received a relatively low score in subjective preference (Table 4). However, interviews with the participants showed both negative and positive feedback as follows.

> P5: "I was uncomfortable to see an asterisk when I entered wrong."
> P7, P8: "The stars are frustrating."
> P10: "The star helped me understand that the text entry was changing the wrong characters into the word I wanted to type."

For the task workloads, NASA-TLX results indicate that there were statistical differences in mental, effort, and total scores, whereas there were no statistical differences in physical, temporal, and performance scores (Table 5). One of the interesting findings is that STAR placed a relatively higher workload than other conditions. We anticipate less workload for STAR because less information is exposed to users. In other words, only the asterisk character is displayed instead of the modified character, so we expected there would be less cognitive overhead. However, it is possible that users made more efforts to infer the characters hidden by the asterisks. In this regard, SHADE placed a relatively lower workload than all the other conditions even though it carries more information than others via color coding for the autocorrected characters. Providing users with the appropriate information about autocorrect not only increases typing speed and accuracy but also reduces the cognitive burden.

For the input method and decoder, users reported that proper nouns (P8, P9, P10) and short words (P12) were hard to type because the VHA decoder could not correct them in some cases. The statistical decoder could not effectively infer proper nouns, which have a very low probability in the language model. It also has a higher miss rate for the short words which have similar probability and input pattern such as 'if' and 'of.' However, many participants reported that they felt more comfortable with the VHA soft keyboard than they thought; they were surprised to see that they were good at typing on an ultrasmall keyboard (P2, P5, P7, P8, P10, P11).

> P5: "I am surprised at how well it worked out on a small screen."
> P8: "This is very smart. I think it would be great for message replying."
> P11: "I am surprised to see that it was recognized even if I took a rough touch of the position."

A participant (P4) with a slight hand tremble also typed without noticeable accuracy degradation (CER = 1.9%). We observed that P3 and P9 had improved the input speed rapidly as the study session went on. We suspect that this rapid improvement is possible because the participants could quickly build the mental model for the decoder.

Through user observation, we found that building the correct conceptual model for the decoder is helpful for improving the accuracy and performance of typing. The SHADE method allows users to understand the state of decoding without frustration. As a result, we made a design decision to adopt the SHADE method as our visual feedback among the four conditions.

VI. EXPERIMENT2: SIMULATIONS ON WORD PREDICTION

We ran simulations to know the maximum keystroke savings (KS) according to the key parameters for the word prediction design. We simulated the optimal user behavior and measured the effects of the three design parameters on KS—the number of contextual words (n), window size—the number of suggested words (k), and the number of candidate words (c). We performed the simulations on the VHA implementation, which is described in section IV.

For the test set, we used MacKenzie's phrases [2] and Enron Mobile phrases [10]. Those two phrase sets are the most widely used to evaluate a text entry method. MacKenzie's phrases consist of 500 sentences, 2,712 words, and 14,307 characters. Enron Mobile phrases consist of 163 sentences, 862 words, and 4,015 characters. For Enron Mobile phrases, we filtered out sentences that contain numbers or punctuation in the middle of a word.

We measured keystroke savings (KS) using the same metric as in the previous study [55]:

$$KS = \frac{keys_{normal} - keys_{with\ prediction}}{keys_{normal}} \times 100\%$$

A. SIMULATION 1: FIXING THE PARAMETER C

We conducted the first simulation to determine the appropriate number of word candidates that yield excellent performance in real-time. We controlled the number of word candidates (c), which is the maximum number of the words generated by the *GeneratePredictCandidates* function in Algorithm 2 (line 2). We measured the Keystroke Saving ratio

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3081173, IEEE Access

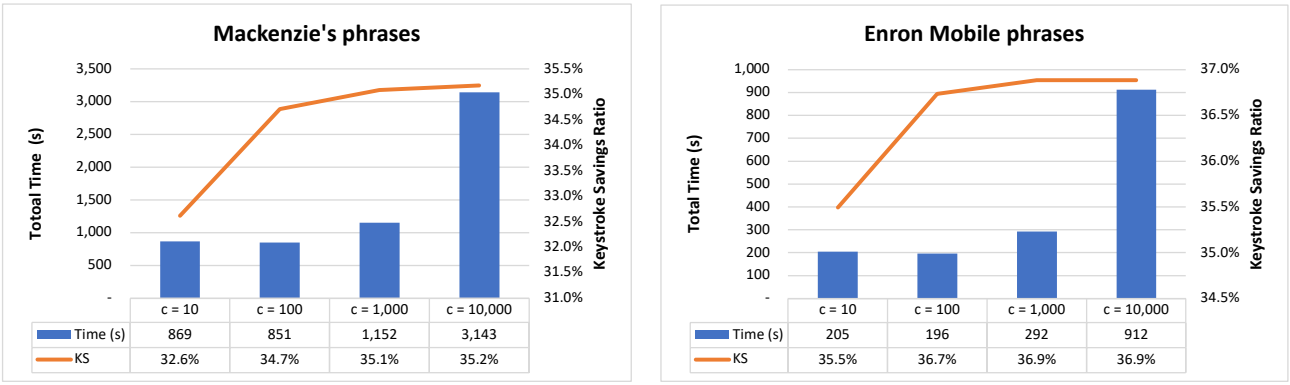


FIGURE 6: Keystroke savings (KS) and its total execution time in seconds are measured along with varying numbers of candidate words c, which is generated by the *GeneratePredictCandidates(P)* function in the prediction algorithm. We used the target device (LG Watch Style) to measure the result against MacKenzie's phrases (500 sentences, 2,712 words, 14,307 characters) and Enron Mobile phrases (163 sentences, 862 words, 4,015 characters).

and total execution time on the real target device–LG Watch Style (LG-W270) smartwatch. We used the same 4-gram language model (n = 4) described in section IV-A. We used the prediction algorithm described in section IV-D. The word prediction algorithm suggested one word (k = 1) for each key input event. When the prediction algorithm suggested the intended word, the user always selects the word and move on to the next one immediately.

Fig. 6 shows the simulation result. In general, the total execution time increases as the number of candidate words for word decoding increases. However, it does not contribute to the keystroke savings at the same rate because low-rank words have little probability to be selected by the prediction algorithm compared to the high-rank words. When the number of maximum candidate words is 10 (c = 10), it takes slightly longer than when c is 100. The main reason is that the hit ratio is lower than the case of c = 100, resulting in more word prediction attempts: 9,640 times when c = 10 and 9,341 times when c = 100 for MacKenzie's phrases; 2,590 times when c = 10 and 2,540 times when c = 100 for Enron Mobile phrases. Based on the experimental results, it is reasonable to set the c value between 100 and 1,000, taking into account the battery efficiency. On average, word prediction was performed with 124 ms in MacKenzie's phrases and 115 ms in Enron Mobile phrases when the number of candidate words (c) is 1,000.

B. SIMULATION 2: COMPARING THE EFFECTS OF THE PARAMETER N AND K

For the second simulation, we controlled the number of context words (n) to query the N-gram language model and the number of word suggestions (k) in the suggestions bar. We fixed the number of candidate words (c) with 1,000. Fig. 7 shows the simulation result.

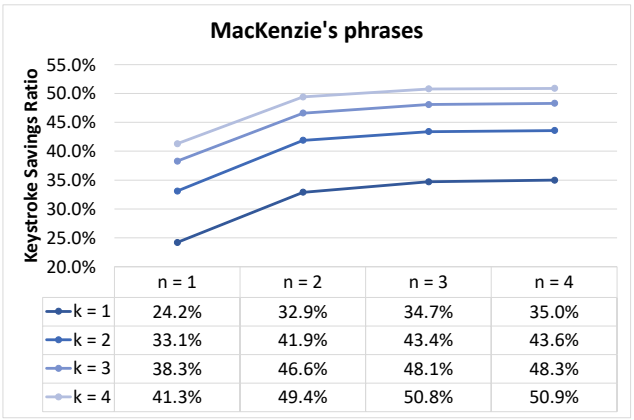
As expected, more suggested words contributed to better keystrokes saving. However, as the number of word suggestions (k) increases, the increase in keystrokes saving becomes smaller and smaller according to the Pareto principle [56]. When using no contextual word (n = 1), the first suggested word contributed 24.2%, while the second, third, and fourth words contributed 8.9%, 5.2%, and 3.0%, respectively, for MacKenzie's phrases. Enron Mobile phrases showed a similar result. The first suggested word contributed 25.5%, while the second, third, and fourth words contributed 7.3%, 5.2%, and 3.9%, respectively. When using three contextual words (n = 4), the first suggested word contributed 35.0%, while the second, third, and fourth words contributed 8.6%, 4.7%, and 2.6%, respectively for MacKenzie's phrases; for Enron Mobile phrases, the first suggested word contributed 36.9%, while the second, third, and fourth words contributed 7.4%, 5.0%, and 2.5%, respectively.

Similarly, when querying the language model, more contextual words contributed to better keystrokes saving. When using one suggested word (k = 1), the no contextual word (n = 1) reported 24.2%, while the one (n = 2), two (n = 3), and three (n = 4) context words reported 32.9%, 34.7%, and 35.0%, respectively for MacKenzie's phrases; for Enron Mobile phrases, the no contextual word (n = 1) reported 25.5%, while the one (n = 2), two (n = 3), and three (n = 4) context words reported 34.1%, 36.7%, and 36.9%, respectively.

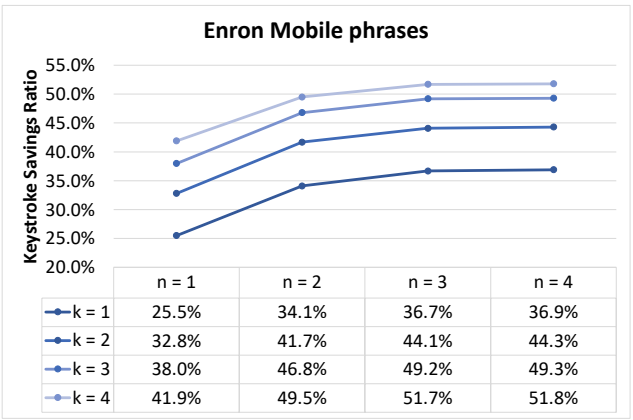
As shown in Fig. 2b, most suggestion bar including GBoard shows two or three words at once. In the suggestion bar, both corrected and predicted words are given to the users. However, providing a suggestion bar increases the number of visual fixation targets from two to three. As a result, users should gaze at the suggestion box in addition to the QWERTY keyboard and the output text box for typing.

Considering that two or three words can appear on a suggestion bar of a smartwatch at a time, we can use one (k = 1) or two (k = 2) slots for the word completion purpose because one slot is already used for the word correction purpose. If we use more than one contextual word based on the underlying 4-gram language model, we can make a similar keystroke saving effect for the single-word selection with

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3081173, IEEE Access



	n = 1	n = 2	n = 3	n = 4
k = 1	24.2%	32.9%	34.7%	35.0%
k = 2	33.1%	41.9%	43.4%	43.6%
k = 3	38.3%	46.6%	48.1%	48.3%
k = 4	41.3%	49.4%	50.8%	50.9%



	n = 1	n = 2	n = 3	n = 4
k = 1	25.5%	34.1%	36.7%	36.9%
k = 2	32.8%	41.7%	44.1%	44.3%
k = 3	38.0%	46.8%	49.2%	49.3%
k = 4	41.9%	49.5%	51.7%	51.8%

FIGURE 7: Keystroke savings (KS) are measured along with varying numbers of contextual words n (based on N-gram model) and numbers of suggested words (window size) k for word completion on two phrase sets—MacKenzie's and Enron Mobile phrases.

the contextual words compared to the multi-words selection without a contextual word, as shown in Fig. 7.

The simulation result with MacKenzie's phrases showed that two contextual words (n = 3) and one suggestion slot (k = 1) showed the keystroke savings ratio as 34.7%, while no contextual word (n = 1) with two suggestion slots (k = 2) showed the keystroke saving ratio as 33.1%. Similarly, the simulation result with Enron Mobile phrases showed that one contextual word (n = 2) and one suggestion slot (k = 1) showed the keystroke savings ratio as 34.1%, while no contextual word (n = 1) with two suggestion slots (k = 2) showed the keystroke saving ratio as 32.8%.

Even though we already simulated the short messages as a test set, messaging on the smartwatch can be more concise than on mobile phones. A single word reply can be more common. However, we can use the conversation history as the contextual words in this case. To support it, the language model also needs to learn the conversation pairs.

All in all, this simulation result leads us to an answer for **Q2**: "**N-gram language model can improve the hit ratio of word prediction large enough to replace the multi-words selection with a single-word one on a smartwatch-sized device.**"

VII. LIMITATION AND FUTURE WORK

Providing accurate decoders and predictions is strongly related to the performance of the autocorrection and suggestion features in the soft keyboard. It is also important for users to build the correct conceptual model of the autocorrection. Effective visual feedback on autocorrection can help users understand the underlying decoder more precisely and thus build a correct conceptual model for the soft keyboard. Our experimental results show that this visual feedback allows users to type more accurately without slowing down the input. It would be an interesting future research direction to investigate whether the effect of visual feedback will be persistent and what kind of visual feedback is preferable for both expert and novice users.

We also found that transcribing some words such as proper nouns, abbreviations, and ambiguous words are error-prone and irritating to users. It is challenging but worth pursuing to research on improving the user experience even in such cases. One solution for texting OOV words is to use a multistep selection method in Table 2, such as ZoomBoard, which reported error rate by 0.1% with 7.6–9.3 wpm when the user turned off the autocorrection mode. Using a personal language model also will be helpful to compensate for OOV words in the general language model and its vocabulary [57]. If users want to correct the decoded output among ambiguous words, we can provide a multi-select user interface. The interface can suggest the probable words based on the likelihood scores of the decoder's candidate word.

VIII. CONCLUSION

We designed and implemented a soft keyboard that runs on ultrasmall touchscreen devices with visual feedback and in situ decoder. The decoder effectively corrected the words that users typed, allowing fast typing (25.3 wpm with CER = 0.9%) even on ultrasmall screens. Users were able to type words more precisely with the help of our visual feedback methods, without sacrificing the input speed. Even though our studies were conducted on smartwatches, we believe that our design and implementation would be of help for other cases where the input channel is noisy, such as accessibility support for those who have severe hand tremors as well as one-handed typing with a thumb.

.

ACKNOWLEDGMENT

We would like to thank the reviews for their suggestions to improve this paper.

SUPPLEMENTARY MATERIALS

REFERENCES

[1] S. Oney, C. Harrison, A. Ogan, and J. Wiese, "Zoomboard: A diminutive qwerty soft keyboard using iterative zooming for ultra-small devices," in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ser. CHI '13. New York, NY, USA: ACM, 2013, pp. 2799–2802. [Online]. Available: http://doi.acm.org/10.1145/2470654.2481387

[2] I. S. MacKenzie and R. W. Soukoreff, "Phrase sets for evaluating text entry techniques," in CHI '03 Extended Abstracts on Human Factors in Computing Systems, ser. CHI EA '03. New York, NY, USA: ACM, 2003, pp. 754–755. [Online]. Available: http://doi.acm.org/10.1145/765891.765971

[3] X. A. Chen, T. Grossman, and G. Fitzmaurice, "Swipeboard: A text entry technique for ultra-small interfaces that supports novice to expert transitions," in Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology, ser. UIST '14. New York, NY, USA: ACM, 2014, pp. 615–620. [Online]. Available: http://doi.acm.org/10.1145/2642918.2647354

[4] J. Hong, S. Heo, P. Isokoski, and G. Lee, "Splitboard: A simple split soft keyboard for wristwatch-sized touch screens," in Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, ser. CHI '15. New York, NY, USA: ACM, 2015, pp. 1233–1236. [Online]. Available: http://doi.acm.org/10.1145/2702123.2702273

[5] L. A. Leiva, A. Sahami, A. Catala, N. Henze, and A. Schmidt, "Text entry on tiny qwerty soft keyboards," in Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, ser. CHI '15. New York, NY, USA: ACM, 2015, pp. 669–678. [Online]. Available: http://doi.acm.org/10.1145/2702123.2702388

[6] T. Shibata, D. Afergan, D. Kong, B. F. Yuksel, I. S. MacKenzie, and R. J. Jacob, "Driftboard: A panning-based text entry technique for ultra-small touchscreens," in Proceedings of the 29th Annual Symposium on User Interface Software and Technology, ser. UIST '16. New York, NY, USA: ACM, 2016, pp. 575–582. [Online]. Available: http://doi.acm.org/10.1145/2984511.2984591

[7] X. Yi, C. Yu, W. Xu, X. Bi, and Y. Shi, "Compass: Rotational keyboard on non-touch smartwatches," in Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, ser. CHI '17. New York, NY, USA: ACM, 2017, pp. 705–715. [Online]. Available: http://doi.acm.org/10.1145/3025453.3025454

[8] M. Gordon, T. Ouyang, and S. Zhai, "Watchwriter: Tap and gesture typing on a smartwatch miniature keyboard with statistical decoding," in Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, ser. CHI '16. New York, NY, USA: ACM, 2016, pp. 3817–3821. [Online]. Available: http://doi.acm.org/10.1145/2858036.2858242

[9] K. Vertanen, H. Memmi, J. Emge, S. Reyal, and P. O. Kristensson, "Velocitap: Investigating fast mobile text entry using sentence-based decoding of touchscreen keyboard input," in Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, ser. CHI '15. New York, NY, USA: ACM, 2015, pp. 659–668. [Online]. Available: http://doi.acm.org/10.1145/2702123.2702135

[10] K. Vertanen and P. O. Kristensson, "A versatile dataset for text entry evaluations based on genuine mobile emails," in Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services, ser. MobileHCI '11. New York, NY, USA: ACM, 2011, pp. 295–298. [Online]. Available: http://doi.acm.org/10.1145/2037373.2037418

[11] K. Vertanen, D. Gaines, C. Fletcher, A. M. Stanage, R. Watling, and P. O. Kristensson, "Velociwatch: Designing and evaluating a virtual keyboard for the input of challenging text," in Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, ser. CHI '19. New York, NY, USA: ACM, 2019, pp. 591:1–591:14. [Online]. Available: http://doi.acm.org/10.1145/3290605.3300821

[12] PHANDROID, "Android wear 2.0 gets a keyboard and can do more without your phone," May 2016. [Online]. Available: http://phandroid.com/2016/05/18/android-wear-2-0-update-details/

[13] K. A. Siek, Y. Rogers, and K. H. Connelly, "Fat finger worries: How older and younger users physically interact with pdas," in Proceedings of the 2005 IFIP TC13 International Conference on Human-Computer Interaction, ser. INTERACT'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 267–280. [Online]. Available: http://dx.doi.org/10.1007/11555261_24

[14] K. B. Min, "A soft keyboard based on statistical decoder for auto-correct and word suggestions running on the android wear platform." September 2019. [Online]. Available: https://github.com/min9079/VisualFeedbackKeyboard

[15] D. Weir, H. Pohl, S. Rogers, K. Vertanen, and P. O. Kristensson, "Uncertain text entry on mobile devices," in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ser. CHI '14. New York, NY, USA: ACM, 2014, pp. 2307–2316. [Online]. Available: http://doi.acm.org/10.1145/2556288.2557412

[16] Microsoft, "How to use the microsoft band keyboard," Video, Feb 2015. [Online]. Available: https://www.youtube.com/watch?v=eE_RFdZhSEo

[17] Apple, "Read and reply to messages with your apple watch." [Online]. Available: https://support.apple.com/en-us/HT205783

[18] S. Ravi, "On-device machine intelligence," February 2017. [Online]. Available: https://research.googleblog.com/2017/02/on-device-machine-intelligence.html

[19] P. Baudisch and G. Chu, "Back-of-device interaction allows creating very small touch devices," in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ser. CHI '09. New York, NY, USA: ACM, 2009, pp. 1923–1932. [Online]. Available: http://doi.acm.org/10.1145/1518701.1518995

[20] H.-S. Yeo, X.-S. Phang, S. J. Castellucci, P. O. Kristensson, and A. Quigley, "Investigating tilt-based gesture keyboard entry for single-handed text entry on large devices," in Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, ser. CHI '17. New York, NY, USA: ACM, 2017, pp. 4194–4202. [Online]. Available: http://doi.acm.org/10.1145/3025453.3025520

[21] A. Gupta and R. Balakrishnan, "Dualkey: Miniature screen text entry via finger identification," in Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, ser. CHI '16. New York, NY, USA: ACM, 2016, pp. 59–70. [Online]. Available: http://doi.acm.org/10.1145/2858036.2858052

[22] J. Goodman, G. Venolia, K. Steury, and C. Parker, "Language modeling for soft keyboards," in Proceedings of the 7th International Conference on Intelligent User Interfaces, ser. IUI '02. New York, NY, USA: ACM, 2002, pp. 194–195. [Online]. Available: http://doi.acm.org/10.1145/502716.502753

[23] D. Weir, S. Rogers, R. Murray-Smith, and M. Löchtefeld, "A user-specific machine learning approach for improving touch accuracy on mobile devices," in Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology, ser. UIST '12. New York, NY, USA: ACM, 2012, pp. 465–476. [Online]. Available: http://doi.acm.org/10.1145/2380116.2380175

[24] X. Yi, C. Yu, W. Shi, and Y. Shi, "Is it too small?: Investigating the performances and preferences of users when typing on tiny qwerty keyboards," International Journal of Human-Computer Studies, vol. 106, pp. 44 – 62, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1071581917300654

[25] X. Bi and S. Zhai, "Bayesian touch: A statistical criterion of target selection with finger touch," in Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology, ser. UIST '13. New York, NY, USA: ACM, 2013, pp. 51–60. [Online]. Available: http://doi.acm.org/10.1145/2501988.2502058

[26] M. D. Kernighan, K. W. Church, and W. A. Gale, "A spelling correction program based on a noisy channel model," in Proceedings of the 13th Conference on Computational Linguistics - Volume 2, ser. COLING '90. Stroudsburg, PA, USA: Association for Computational Linguistics, 1990, pp. 205–210. [Online]. Available: https://doi.org/10.3115/997939.997975

[27] E. Mays, F. J. Damerau, and R. L. Mercer, "Context based spelling correction," Inf. Process. Manage., vol. 27, no. 5, pp. 517–522, Sep. 1991. [Online]. Available: http://dx.doi.org/10.1016/0306-4573(91)90066-U

[28] N. Garay-Vitoria and J. Abascal, "Text prediction systems: A survey," Univers. Access Inf. Soc., vol. 4, no. 3, pp. 188–203, Feb. 2006. [Online]. Available: https://doi.org/10.1007/s10209-005-0005-9

[29] P. Quinn and S. Zhai, "A cost-benefit study of text entry suggestion interaction," in Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, ser. CHI '16. New York, NY, USA: ACM, 2016, pp. 83–88. [Online]. Available: http://doi.acm.org/10.1145/2858036.2858305

[30] N. Banovic, V. Rao, A. Saravanan, A. K. Dey, and J. Mankoff, "Quantifying aversion to costly typing errors in expert mobile text entry," in Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, ser. CHI '17. New York, NY, USA: ACM, 2017, pp. 4229–4241. [Online]. Available: http://doi.acm.org/10.1145/3025453.3025695

[31] A. S. Arif, S. Kim, W. Stuerzlinger, G. Lee, and A. Mazalek, "Evaluation of a smart-restorable backspace technique to facilitate text entry error correction," in Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, ser. CHI '16. New York, NY, USA: ACM, 2016, pp. 5151–5162. [Online]. Available: http://doi.acm.org/10.1145/2858036.2858407

[32] NUANCE, "Swype," retrieved September 10, 2017 from http://www.swype.com.

[33] Google, "Android ndk." [Online]. Available: https://developer.android.com/ndk

[34] Twitter, "Filter realtime tweets." [Online]. Available: https://developer.twitter.com/en/docs/tweets/filter-realtime/overview

[35] K. Vertanen and P. O. Kristensson, "The imagination of crowds: Conversational aac language modeling using crowdsourcing and large data sources," in Proceedings of the Conference on Empirical Methods in Natural Language Processing, ser. EMNLP '11. Stroudsburg, PA, USA: Association for Computational Linguistics, 2011, pp. 700–711. [Online]. Available: http://dl.acm.org/citation.cfm?id=2145432.2145514

[36] A. Stolcke, "Srilm - an extensible language modeling toolkit." in Proceedings of International Conference on Spoken Language Processing, vol. 2, 2002, pp. 901–904.

[37] G. Ward, "Moby word lists by grady ward," June 1996. [Online]. Available: http://www.gutenberg.org/ebooks/3201

[38] I. H. Witten and T. C. Bell, "The zero-frequency problem: estimating the probabilities of novel events in adaptive text compression," IEEE Transactions on Information Theory, vol. 37, no. 4, pp. 1085–1094, Jul 1991.

[39] S. F. Chen and J. Goodman, "An empirical study of smoothing techniques for language modeling," in Proceedings of the 34th Annual Meeting on Association for Computational Linguistics, ser. ACL '96. Stroudsburg, PA, USA: Association for Computational Linguistics, 1996, pp. 310–318. [Online]. Available: https://doi.org/10.3115/981863.981904

[40] C. Chelba, T. Brants, W. Neveitt, and P. Xu, "Study on interaction between entropy pruning and kneser-ney smoothing." in INTERSPEECH, 2010, pp. 2422–2425.

[41] K. Heafield, "Kenlm: Faster and smaller language model queries," in Proceedings of the Sixth Workshop on Statistical Machine Translation, ser. WMT '11. Stroudsburg, PA, USA: Association for Computational Linguistics, 2011, pp. 187–197. [Online]. Available: http://dl.acm.org/citation.cfm?id=2132960.2132986

[42] Wikipedia, "List of common misspellings/for machines," 2013. [Online]. Available: https://en.wikipedia.org/wiki/Wikipedia:Lists_of_common_misspellings/For_machines

[43] R. Mitton, "Corpora of misspellings for download." [Online]. Available: http://www.dcs.bbk.ac.uk/~ROGER/corpora.html

[44] aspell, "Test data." [Online]. Available: http://aspell.net/test/cur/batch0.tab

[45] Y. Baba and H. Suzuki, "How are spelling errors generated and corrected?: A study of corrected and uncorrected spelling errors using keystroke logs," in Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2, ser. ACL '12. Stroudsburg, PA, USA: Association for Computational Linguistics, 2012, pp. 373–377. [Online]. Available: http://dl.acm.org/citation.cfm?id=2390665.2390749

[46] K. Kukich, "Techniques for automatically correcting words in text," ACM Comput. Surv., vol. 24, no. 4, pp. 377–439, Dec. 1992. [Online]. Available: http://doi.acm.org/10.1145/146370.146380

[47] X. Bi and S. Zhai, "Predicting finger-touch accuracy based on the dual gaussian distribution model," in Proceedings of the 29th Annual Symposium on User Interface Software and Technology, ser. UIST '16. New York, NY, USA: ACM, 2016, pp. 313–319. [Online]. Available: http://doi.acm.org/10.1145/2984511.2984546

[48] E. Brill and R. C. Moore, "An improved error model for noisy channel spelling correction," in Proceedings of the 38th Annual Meeting on Association for Computational Linguistics, ser. ACL '00. Stroudsburg, PA, USA: Association for Computational Linguistics, 2000, pp. 286–293. [Online]. Available: https://doi.org/10.3115/1075218.1075255

[49] L. Philips, "Hanging on the metaphone," Computer Language Magazine, vol. 7, no. 12, pp. 39–44, December 1990.

[50] "Lg watch style (lg-w270)," 2017. [Online]. Available: https://www.lg.com/us/smart-watches/lg-W270-Titanium-style/

[51] "Snapdragon wear 2100 processor," 2017. [Online]. Available: https://www.qualcomm.com/products/snapdragon/processors/wear-2100

[52] S. G. Hart and L. E. Staveland, "Development of nasa-tlx (task load index): Results of empirical and theoretical research," in Human Mental Workload, ser. Advances in Psychology, P. A. Hancock and N. Meshkati, Eds. North-Holland, 1988, vol. 52, no. Supplement C, pp. 139 – 183. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0166411508623869

[53] I. S. MacKenzie, "A note on calculating text entry speed," March 2015. [Online]. Available: http://www.yorku.ca/mack/RN-TextEntrySpeed.html

[54] R. W. Soukoreff and I. S. MacKenzie, "Metrics for text entry research: An evaluation of msd and kspc, and a new unified error metric," in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ser. CHI '03. New York, NY, USA: ACM, 2003, pp. 113–120. [Online]. Available: http://doi.acm.org/10.1145/642611.642632

[55] K. Trnka and K. F. McCoy, "Evaluating word prediction: Framing keystroke savings," in Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers, ser. HLT-Short '08. Stroudsburg, PA, USA: Association for Computational Linguistics, 2008, pp. 261–264. [Online]. Available: http://dl.acm.org/citation.cfm?id=1557690.1557766

[56] M. Newman, "Power laws, pareto distributions and zipf's law," Contemporary Physics, vol. 46, no. 5, pp. 323–351, 2005. [Online]. Available: https://doi.org/10.1080/00107510500052444

[57] A. Fowler, K. Partridge, C. Chelba, X. Bi, T. Ouyang, and S. Zhai, "Effects of language modeling and its personalization on touchscreen typing performance," in Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, ser. CHI '15. New York, NY, USA: ACM, 2015, pp. 649–658. [Online]. Available: http://doi.acm.org/10.1145/2702123.2702503

KU BONG MIN received the B.S. and M.S. degree in computer science and engineering from Seoul National University, South Korea, in 2001.

He is a Research Fellow at LG Electronics, South Korea. He is also currently working toward a Ph.D. degree with the Department of Computer Science and Engineering, Seoul National University, South Korea. His research interests include human-computer interaction, information visualization, and interaction design for consumer electronics.

JINWOOK SEO (Senior Member, IEEE) received the B.S. and M.S. degrees in computer science from Seoul National University and the Ph.D. degree in computer science from the Human-Computer Interaction Lab, University of Maryland, College Park, USA, in 2005.

He is currently a Professor at the Department of Computer Science and Engineering, Seoul National University, South Korea. His research interests include human–computer interaction, interaction design, information visualization, visual analytics, and health informatics.