The Impact of Word, Multiple Word, and Sentence Input on Virtual Keyboard Decoding Performance

Keith Vertanen[†], **Crystal Fletcher**[†], **Dylan Gaines**[†], **Jacob Gould**[†], **Per Ola Kristensson**[‡]

[†]Department of Computer Science
Michigan Technological University
Houghton, Michigan, USA
{vertanen|tafletch|dcgaines|jcgould}@mtu.edu

[‡]Department of Engineering
University of Cambridge
Cambridge, United Kingdom
pok21@cam.ac.uk

ABSTRACT

Entering text on non-desktop computing devices is often done via an onscreen virtual keyboard. Input on such keyboards normally consists of a sequence of noisy tap events that specify some amount of text, most commonly a single word. But is single word-at-a-time entry the best choice? This paper compares user performance and recognition accuracy of word-at-a-time, phrase-at-a-time, and sentence-at-a-time text entry on a smartwatch keyboard. We evaluate the impact of differing amounts of input in both text copy and free composition tasks. We found providing input of an entire sentence significantly improved entry rates from 26 wpm to 32 wpm while keeping character error rates below 4%. In offline experiments with more processing power and memory, sentence input was recognized with a much lower 2.0% error rate. Our findings suggest virtual keyboards can enhance performance by encouraging users to provide more input per recognition event.

ACM Classification Keywords

H.5.2 Information interfaces and presentation: Input devices and strategies

Author Keywords

Text entry; virtual keyboard; decoder; smartwatch

INTRODUCTION

The auto-correcting on-screen virtual keyboard is a ubiquitous text entry method for mobile devices. Users typically interact with a touchscreen keyboard by providing input one word at a time. After a word is specified, a statistical decoder searches for the most probable word given the observations recorded by the keyboard. The oldest and most common approach is to enter a word via a sequence of discrete taps [7]. Alternatively a word can be continuous gesture over its letters on the keyboard [15, 43]. The best word hypothesis is then output with the user either carrying on to the next word, or taking some corrective action if the word was misrecognized.

As noted by Vertanen et al. [34], this word-at-a-time approach could slow users as they may spend time monitoring each word recognition result. Further, the word-at-a-time approach provides the recognizer with only one word's worth of noisy observations from which to infer the user's intended text. It is plausible the auto-correct algorithm may be able to make more accurate guesses if it can guess a sequence of words since later words can influence the search for the most likely hypothesis that hopefully matches the user's intention.

We explore allowing users to *modulate* the amount of input given to the auto-correct decoder. We are interested in investigating if such modulation can reduce error rate or increase entry rate. We explore allowing users to control the amount of words sent to the auto-correct decoder. We hypothesize that sending several words to the decoder may both increase entry rate by reducing the number of motor actions and improve decoder accuracy by providing more information about the user's intended word sequence. We find that such modulation does indeed increase entry rate significantly and in a follow-up study with both a text copy task and a composition task we find that users prefer either a word-at-a-time or a sentence-at-a-time approach. In addition, we conduct recognition experiments to investigate under what circumstances providing more input to the decoder actually improves performance.

Mobile text entry methods are increasingly being designed for hardware relying on uncertain sensing of user input. One example is optical see-through head-mounted displays, which sense the user's hand or finger location using body-fixed depth sensing. Another example are small wearable devices, such as pendants or smartwatches, where the form factor makes precise direct control noisy, even though the sensor itself may be accurate. We propose using a smartwatch as a testbed for investigating virtual keyboard decoding as it provides sufficiently challenging noisy tap observations from users due to the very small form factor. We adapt a state-of-the-art touchscreen keyboard decoder [34] so that it can run on a smartwatch and decode with high accuracy and low latency. This commercial-grade testbed ensures validity in the experiments and relevance for future hardware platforms which may have to handle noisy sensing, noisy user control, or both.

CHI 2018, April 21–26, 2018, Montreal, QC, Canada

ACM ISBN 978-1-4503-5620-6/18/04…$15.00
https://doi.org/10.1145/3173574.3174200

RELATED WORK

Text entry has attracted considerable research interest (see [18, 44, 20] for surveys and overviews). In general, the objective of text entry is to allow users to write as quickly and as accurately as possible, although in recent years it has been proposed to target text entry rates that achieve the inviscid text entry rate, that is, the text entry rate which is rate-limited by users' creativity [14]. Importantly, many text entry methods proposed in the literature have failed to achieve text entry rates that surpass mainstream mobile text entry methods [14, 12].

Intelligent text entry methods infer [7] or predict [6] users' intended text from noisy input [11]. As mobile devices transitioned into capacitive touchscreen devices and processing power increased, such text entry solutions have become ubiquitous in commercial mobile keyboard systems. Typically, the user either taps on a touchscreen keyboard or uses a gesture keyboard, which allow users to write text by articulating word gestures across a touchscreen keyboard [41, 15, 10, 43]. By allowing for noisy user input, intelligent text entry methods are more likely to achieve higher entry rates [11, 12].

An orthogonal research dimension is keyboard optimization. Early attempts focused on minimizing average movement time between keys [21, 39]. Later approaches considered additional factors and investigated alphabetical ordering [45] in order to make it easier for users to learn an unfamiliar keyboard layout (see Zhai et al. [40] for an early survey and Bi et al. [3] for further work in this direction). This was improved further by for example considering optimization for thumb-typing ergonomics [22], multiple languages [4], and auto-correct [2] and gesture keyboard performance [42, 24, 25].

Decoding

In this paper we focus on inferring users' intended text from noisy tap data on virtual keyboards. When this process is carried out using a probabilistic method it is known as decoding. At a very high-level, the objective is to identify the most probable character sequence C^* given a set of observations (for instance, key presses) O:

$$C^* = \arg\max_{C \in \mathbb{C}} \left[P(O|C)P(C) \right], \tag{1}$$

where $\mathbb{C}$ is the set of all permissible character sequences in the system. $P(O|C)$ is the keyboard touch model and $P(C)$ is the language model. Calculating Equation 1 necessitates a decoding process. Typically this decoding process is carried out in a similar vein as classic speech recognition decoding [38, 23, 9]. Goodman et al. [7] proposed a decoder that used a keyboard touch model based on two-dimensional Gaussian distributions for each key and a character-based language model to substitute touch point observations for letter keys. Kristensson and Zhai [16] proposed an alternative approach which relied on geometric pattern matching.

Vertanen, et al. [34] investigated sentence-at-a-time touchscreen input using a statistical decoder named VelociTap. Vertanen, et al. tested delimiting words via a space key, by swiping to the right, or by inferring space automatically. Inferring spaces was found to be the fastest while using a space key was found to be the most accurate. VelociTap accurately recognized sentences on keyboards down to the size of a smartwatch (but in [34] was tested on a phone). Recognition was proxied to a powerful desktop computer. While Vertanen, et al. conjectured that a sentence-at-a-time approach might be faster than word-at-a-time input, this was not explicitly tested.

A fundamental problem with decoding-based keyboard input is failure of the algorithm to identify the user's intended word, sometimes called the "auto-correct trap" [35]. Weir et al. [35] found it was possible to reduce error rates in keyboard decoding by allowing users to modulate their certainty by pressure. The harder the user pressed on a key, the more the decoder would believe the touch point observation belonged to the pressed key. In addition, Weir et al. [35] found that it was possible to achieve a small gain in accuracy by using a touch point model based on Gaussian Process regression.

Smartwatch Text Entry

Smartwatches have recently gained popularity and as a consequence, researchers have investigated methods of efficiently entering text on them, see Arif and Mazalek [1] for a recent and extensive survey of smartwatch text entry.

The decoding-based keyboard VelociTap [34] was the first to suggest and demonstrate the viability of typing on a full-key QWERTY keyboard on a smartwatch aided by a decoder. Later, Gordon et al. [8] presented a smartwatch keyboard based on statistical decoding that supported tap input, gesture-keyboard input, and word prediction. In a user study, participants wrote at over 22 wpm with a low error rate.

In a more recent study, Yi et al. [37] collected touch data on a variety of tiny touchscreen keyboard sizes. They used the data to create a touch model which was combined with a language model to produce a touchscreen keyboard decoder. User studies demonstrated fast entry rates of over 27 wpm with low error rates. Finally, Turner at al. [27] used the commercial gesture keyboard Swype to investigate tap and trace input on a smartwatch. In a user study, participants wrote at 37 wpm using the trace method and 27 wpm using the tap method.

Text Entry Evaluation

Text entry methods are typically evaluated in controlled experiments using a transcription-task where users copy stimulus sentences (or phrases) as quickly and as accurately as possible. MacKenzie and Soukoreff [19] suggested stimulus phrases should be standardized across studies and released a phrase set for this purpose. Vertanen and Kristensson [31] later released an alternative phrase set based on genuine mobile email texts, which are more representative of text users actually write on mobile devices, although the performance between phrase sets did not differ significantly in a crowdsourced user study [13].

Later research has investigated further ways of sampling representative phrases based on criteria such as word clarity [36]. While a transcription task ensures high internal validity, it inevitably does not fully model actual typing, which includes the cognitive effort of composing the text. A complementary evaluation strategy is therefore to use a composition task, in which users are asked to compose messages [32].

SYSTEM

Our system is compromised of two parts, a core recognition component and a virtual keyboard interface designed for a smartwatch. We describe each part in turn.

Noisy Tap Recognizer

Our input method uses a recognition-based approach that attempts to infer a user's intended text from a noisy sequence of tap locations. Our method is based on the VelociTap touchscreen decoder [34]. VelociTap uses a keyboard touch model that assigns a probability to all possible keyboard characters for every tap. The keyboard model assigns probabilities based on a tap's location and two-dimensional, axis-aligned Gaussians centered at each key center. The x- and y-variances are configurable but tied between all keys.

Every tap sequence produces many recognition hypotheses. The probability of these hypotheses under the keyboard model is combined with their probability under a character and word language model. VelociTap supports the insertion of all possible characters anywhere in the input sequence. Insertions incur a configurable penalty. In order to better support spaceless multi-word input, VelociTap also has a separate space insertion penalty. Taps can also be deleted without generating any output. Deletions incur a configurable penalty.

VelociTap was originally designed for sentence decoding with words separated by a space bar, a right swipe gesture, or with no explicit separation. We modified VelociTap to recognize a single or multiple words from a sequence of virtual keyboard tap locations. This involved allowing surrounding text to influence the recognition process (e.g. three taps are more likely to be "day" if the previous text is "have a good").

During the decoder's search, a character sequence may have zero or more spaces inferred somewhere in the sequence. This delimits the sequence into one or more words. The character sequence is scored by a character language model and the delimited word sequence is scored by a word language model. Any words that are out-of-vocabulary (OOV) are replaced by an unknown word prior to scoring by the word language model and also assessed a configurable OOV penalty.

We tuned the configurable parameters of the decoder on data collected from the studies reported in [34]. We further fine tuned them on development data recorded by three of the authors on the watch used in the study reported here.

We trained our language models on billions of words from twitter, Usenet, blogs, social media, movie subtitles, and forum posts. Our character 12-gram model used Witten-Bell smoothing and had a vocabulary of `a-z`, the characters `',.?!` and a space pseudo-word. Our word 4-gram model used modified Kneser-Ney smoothing and a vocabulary of 100 K words[1].

In order to fit within the memory limits of a watch, we heavily pruned our models using entropy pruning [26]. During pruning we used lower order Good-Turing smoothed models [5]. The character model had 588 K N-grams and a gzipped ARPA text format size of 5.5 MB. The word model had 766 K N-grams and a size of 6.3 MB.

[1] http://keithv.com/data/vocab_100k.txt

Figure 1. Example of writing "i am waiting for aiden". The user typed "waiting for" as a multi-word input (left). Swiping right recognized the words. Uf a finger is down, the nearest key appears in a large font (middle). After typing the remaining letters, the user obtains "aiden" (right).

In our previous work [34], we forwarded mobile device input to a desktop computer for recognition. In this work we performed recognition on-device. To better support on-device recognition, we improved our decoder's efficiency by reducing how many objects were created and destroyed and by adding binary serialization of file assets such as the language models.

Smartwatch Keyboard

We conducted all experiments using a Android Wear app we developed for the Sony Smartwatch 3. This watch has a 29 mm square touchscreen with a resolution of 320 × 320. It has a 4-core 1.2 GHz ARM CPU and 512 MB of memory. Recognition was performed locally on the watch.

Our QWERTY keyboard is 29 mm × 13 mm (Figure 1). Our keyboard has the letters `a-z` with apostrophe in the lower right. We use white labels at the center of each key with no key outlines. This yields an effective key size of 2.9 mm × 4.3 mm. To be considered a valid tap, the touch down and up locations had to be in the keyboard area or just slightly above (within 3 mm). The device vibrates for 50 ms after each tap or swipe.

The area above the keyboard shows previously recognized text as well as the nearest keys for the current unrecognized input. When a user's finger is in contact with the screen, the label of the nearest key is displayed in a large font alpha channeled over the result area. This allows the user to reposition their finger despite the visual occlusion caused by their finger.

Users request recognition by swiping to the right anywhere on the screen. When recognition is in progress, the screen turns green and input is ignored until recognition completes. Recognition results are added to the text area above the keyboard. Swiping to the left during input deletes the previous tap's nearest key text and removes it from the observations sent to the decoder. Swipes 3 mm or longer were classified as up, down, left, or right by the angle from the starting position.

USER STUDY: INPUT AMOUNT

Our study investigates the human and recognition performance impacts of word, multiple word, and sentence input. Our goal is to fairly compare the entry and error rate potential of, and user preference for, different input sizes. Our intention is not to propose or beat an existing method. Rather, we want to find out whether there exist human and recognition accuracy benefits to larger input sizes. While larger input has been

supported in commercial keyboards for a number of years, no study has investigated its performance characteristics.

In Experiment 1 we exposed users to all three input amounts by having them enter memorable sentences using each approach. Once users had experienced the different input amounts, in Experiment 2 we investigated what input behavior they would use in practice. In Experiment 2 we also wanted to understand if users' behavior was affected by the experimental task. Typically in text entry experiments participants are given the task of copying memorable text. In Experiment 2 we compare this with a composition task in which participants invented text.

Correction Features

In both experiments, a left swipe deleted the previous tap during entry, but prior to recognition. We allowed this since: 1) it did not favor any condition, 2) it was quick to perform, 3) it provided a measure to compare input behavior in Experiment 2's copy and composition tasks, and 4) it is a viable correction method for a range of devices and use scenarios.

We decided not to allow users to backspace or otherwise correct recognition errors. Further we did not provide a suggestion bar that might offer recognition alternatives or word completions. We did this for the following reasons:

1. While backspacing errors and suggestion bars provide a familiar correction method for word input, they may not constitute the best approaches for multiple word or sentence input. How to design correction interfaces to best support larger input sizes is an open question. There are a variety of ways this might be done, e.g. selecting from a word confusion network [29] or automatic positioning a correction within the original text [28, 30]. We designed our study to test the role of input size in isolation. If we had varied both input size and the correction interface in each condition, it would be difficult to determine how much each contributed to any difference.

2. We wanted to reduce variability not associated with the input size. In our experience, participants in text entry studies often work to fix most errors leading to highly variable entry times, especially when errors cascade. By removing post-recognition correction, we could more accurately measure participants' input speed.

3. While these features are common on today's virtual keyboards, future devices or challenging use scenarios may preclude their use, e.g. touch interaction may be on-body, on uninstrumented surfaces, or used while visually attending to your environment. We argue informing the design of future devices or use scenarios is best done by understanding how interaction building blocks work in isolation rather than when combined into one particular interface design.

Participants, Metrics, and Study Tasks

We recruited 24 participants via convenience sampling. Participants took part in a one-hour session and were paid $10. Participants were aged 18 to 21 (mean 18.5). 13 identified as male, 7 as female, and the rest did not answer. 20 were right handed. All participants strongly agreed that they were fluent English speakers. 18 participants had never used a smartwatch, 3 used one occasionally, and 3 used one frequently.

We measured input errors using *Character Error Rate (CER)*. CER is the number of character insertions, deletions, and substitutions required to change the recognized text into the reference text, divided by the characters in the reference (multiplied by 100). We also wanted to measure the error rate prior to any recognition. We achieved this by calculating what we call the *literal CER*. The literal CER is found by first identifying the nearest key to each tap observation (touch point) from the user and then computing the CER between the stimulus text and the reference text, ignoring any spaces in the text.

We measured entry rate using *words-per-minute (wpm)* with a word being five characters including space. We calculated entry time from the first tap on the keyboard screen to when the final recognition result was displayed.

All participants wore the watch on their non-dominant hand. Participants were asked to type using the index finger of their dominant hand. Similar to [17], we measured the width of participants' index finger at the base of their fingernail with a digital caliper. Participants were asked to rest their watch-wearing arm on the top of the desk they were seated at.

Participants first filled out a two-page questionnaire asking about their previous experience with desktop and mobile text input. After each condition in all experiments, participants filled out a one-page questionnaire. This questionnaire asked them to rate how quickly and accurately they felt they entered text in that condition. It also asked for open comments.

In all but one condition, participants entered memorable sentences from the `mem1-5` sets in the Enron mobile data set [31]. Due to the small screen size, we limited sentences to those with six or fewer words. All sentences were shown in lowercase with no punctuation aside from apostrophes. Each participant received sentences chosen at random from a set of 143 sentences. No sentence was given to a participant more than once. Participants could spend as long as they wanted memorizing a sentence. The sentence disappeared and could not be referred to again once keyboard input began. In one condition, participants composed their own mobile messages. We used the composition procedure described in [33].

Unless otherwise stated, we tested for significance using a repeated measures analysis of variance. For significant main effects, we used Bonferroni corrected post-hoc tests. In some cases, we report results with $\pm$ values giving 95% confidence intervals (CIs) of the mean calculated using a t-distribution.

EXPERIMENT 1: FIXED INPUT AMOUNT

This was a within-subject experiment with three counterbalanced conditions:

- **WORD** – Participants were asked to use a word-at-a-time entry approach.
- **TWOWORD** – Participants were asked to enter text two words at a time. This was done by entering both words without any input signifying the space between the words. In the case of sentences with an odd number of words, participants were told to enter the final word by itself. We conjecture in real-world use, users may segment text into multiple word chunks based on text aspects such as syntax

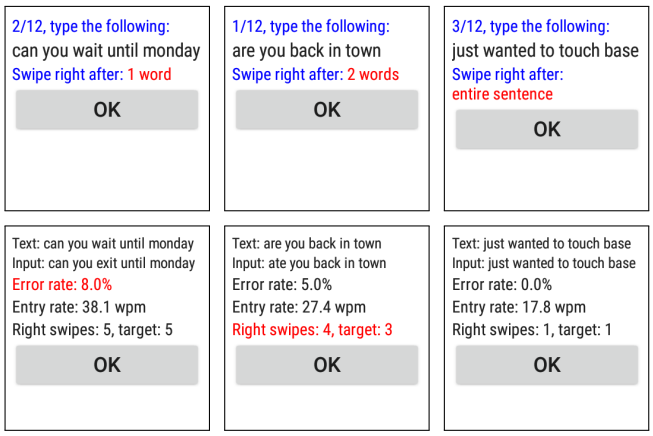

Figure 2. Prompts before each task (top) and post-input feedback (bottom) for the three conditions in Experiment 1: WORD (left), TWOWORD (middle), and SENTENCE (right).

or semantics. However for purposes of this experiment, we decided on an explicit two-word input strategy to provide easy-to-follow and unambiguous instructions.

- **SENTENCE** – Participants were asked to enter the entire sentence before requesting recognition. As with TWOWORD, no input action was required to denote spaces.

Participants used the smartwatch keyboard described previously. All conditions used the same keyboard, recognition setup, and text copy task. The only difference between conditions was the instructions given by the experimenter and by the app about how to enter text (Figure 2 top). We told participants to carry on after any recognition errors with a goal of entering the text "quickly and accurately".

After each sentence, participants were shown the original sentence, their final input, and their error and entry rate on the sentence (Figure 2 bottom). We also showed them the number of right swipe recognition events they requested and the target value for the given condition and sentence. If the error rate was above 5% CER or the number of right swipes was incorrect, this text was shown in red and the watch vibrated twice. After completing each condition, participants were shown their average error and entry rate for all sentences in the condition.

Participants first completed four practice sentences in each of the three conditions. After all practice sentences, participants completed 12 sentences in each of the three conditions. We only analyzed data from the non-practice tasks.

Results

Figure 3 shows boxplots of the main results. Table 1 provides numeric results and statistical test details.

Participants became faster as the size of their input increased. Entry rate increased from 26 wpm in WORD, to 29 wpm in TWOWORD, to 32 wpm in SENTENCE. These differences were all statistically significant (Table 1). Entry time included the time it took to recognize input. The impact of recognition time on entry rate was negligible, recognition took 0.054 s in WORD, 0.084 s in TWOWORD, and 0.23 s in SENTENCE.

Error rate varied depending on the amount of input per recognition. CER was lowest in TWOWORD at 2.7%, followed by WORD 3.1%, and finally SENTENCE 3.9%. These differences were not statistically significant (Table 1).

The literal CER (before any recognition) was lowest in SENTENCE at 15.3%, followed by TWOWORD at 17.6%, and finally WORD at 19.4%. Only the difference between SENTENCE and WORD was statistically significant (Table 1). These results suggest an advantage of larger input sizes is that users may more accurately target letters when their input is continuous and not broken by delimiting word boundaries or checking recognition results.

The literal CER was highly variable between participants. As expected, most participants had a high error rate on the small keyboard. However, four participants had a low literal CER of less than 3% (averaged over all conditions). One participant actually correctly targeted every single letter in all conditions.

Participants could left swipe to backspace any erroneous taps. In all conditions the number of backspaces per final output character was low: WORD at 0.066, TWOWORD at 0.054, and SENTENCE at 0.067. These differences were not statistically significant (Table 1). This indicates that despite often seeing incorrect letters in the pending input, participants tended to trust that the auto-correct algorithm would bail them out.

Based on the condition and sentence, we calculated how often participants input the correct amount. SENTENCE was the easiest with correct behavior on 99% of tasks. WORD was next at 98% correct behavior. TWOWORD was more difficult at 94%. Only the difference between TWOWORD and SENTENCE was statistically significant (Table 1). The difficulty of TWOWORD was confirmed by participants' comments (Table 2).

After each condition, participants rated statements on a 7-point Likert scale (1=strongly disagree, 7=strongly agree). The mean rating for the statement "I thought I entered text *quickly*" was 5.25 in WORD, 5.58 in TWOWORD, and 5.67 in SENTENCE (Figure 4 left). This difference was not statistically significant (Friedman's test, $\chi^2(2) = 2.14$, $p = 0.34$). The mean rating for the statement "I thought I entered text *accurately*" was 5.25 in WORD, 5.08 in TWOWORD, and 4.54 in SENTENCE (Figure 4 right). This difference was not statistically significant (Friedman's test, $\chi^2(2) = 3.92$, $p = 0.14$).

As shown in Table 2, participants had both positive and negative things to say about each strategy. This indicates we may want to design input methods that provide participants flexibility in how much input they provide for each recognition.

EXPERIMENT 2: FLEXIBLE INPUT AMOUNT

After a short break, participants proceeded to Experiment 2. We explained to participants they were now free to use whatever input strategy they liked and that they could switch freely between them. This was a within-subject experiment with two counterbalanced conditions:

- **COPY** – Participants entered memorable sentences.
- **COMPOSE** – Participants were asked to invent a fictitious message that they might write on a mobile device.

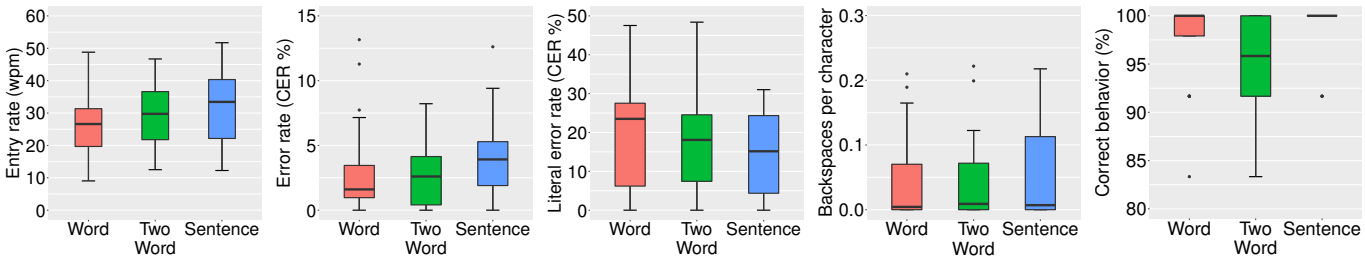

Figure 3. Entry rate, error rate (after recognition), literal error rate (before recognition), backspaces/character, and correct behavior in Experiment 1.

	Entry rate (wpm)	Error rate (CER %)	Literal error rate (CER %)
WORD	26.2 ± 3.6 [9.0, 48.8]	3.1 ± 1.5 [0.8, 13.2]	19.4 ± 5.9 [0.0, 47.5]
TWOWORD	28.9 ± 3.8 [12.5, 46.7]	2.7 ± 1.0 [0.0, 8.2]	17.6 ± 5.7 [0.0, 48.4]
SENTENCE	31.8 ± 4.6 [12.2, 51.7]	3.9 ± 1.2 [0.0, 12.6]	15.3 ± 4.5 [0.0, 31.0]
ANOVA	$F_{2,46} = 24.8$, $\eta_p^2 = 0.52$, $p < .001$	$F_{2,46} = 1.7$, $\eta_p^2 = 0.07$, $p = 0.199$	$F_{2,46} = 6.5$, $\eta_p^2 = 0.22$, $p < .01$
Post-hoc	WORD < TWOWORD, $p < .001$	Not applicable	SENTENCE < WORD, $p < .01$
	TWOWORD < SENTENCE, $p < .01$		TWOWORD ≈ SENTENCE, $p = 0.177$
	WORD < SENTENCE, $p < .001$		TWOWORD ≈ WORD, $p = 0.361$

	Backspaces per character	Correct behavior (%)
WORD	0.065 ± 0.054 [0.00, 0.57]	97.6 ± 1.9 [83.3, 100.0]
TWOWORD	0.054 ± 0.038 [0.00, 0.36]	93.8 ± 3.5 [66.7, 100.0]
SENTENCE	0.067 ± 0.048 [0.00, 0.49]	99.0 ± 1.2 [91.7, 100.0]
ANOVA	$F_{2,46} = 1.2$, $\eta_p^2 = 0.05$, $p = 0.309$	$F_{2,46} = 6.1$, $\eta_p^2 = 0.21$, $p < .01$
Post-hoc	Not applicable	TWOWORD < SENTENCE, $p < .05$
		TWOWORD ≈ WORD, $p = 0.055$
		SENTENCE ≈ WORD, $p = 0.770$

Table 1. Results from Experiment 1. Results formatted as: mean ± 95% CI [min, max]. The bottom section of each table shows the repeated measures ANOVA statistical test for each dependent variable. For significant omnibus tests, we show pairwise post-hoc tests (Bonferroni corrected).

WORD
+ "Less stress to spell the words correctly"
+ "Swiping right after every word felt natural"
– "Slow due to large amounts of swiping"
– "Swiping after each word slowed me down"
TWOWORD
+ "Easier than single words and full sentences"
+ "This version worked the best for me"
– "Hard to remember to type two words then swipe"
– "Swiping every two words interrupted flow"
SENTENCE
+ "Fast to type as there is less right swiping"
+ "Felt extremely fast"
– "It was hard to make sure I entered the letters correctly when every word was mixed"
– "Harder for the watch to correct me"

Table 2. Selected positive and negative comments provided by participants about each condition in Experiment 1.

The procedure and interface was almost identical to Experiment 1. We changed the task prompt screen and post-input review screen to reflect the composition task and to indicate they were free to choose their input amount (Figure 5). Participants first completed four practice tasks in both conditions. This was followed by 12 evaluation tasks in both conditions. We only analyzed data from the non-practice tasks.

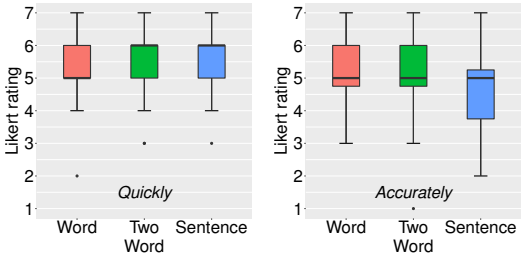

Figure 4. Subjective ratings about whether participants felt they entered text quickly (left) or accurately (right) in Experiment 1.

Results

Figure 6 shows boxplots of the main results. Table 3 provides numeric results and statistical test details.

Entry rates were faster in COPY at 33.4 wpm compared to COMPOSE at 30.0 wpm. This difference was statistically significant (Table 3) Recognition time was 0.11 s in both conditions. Participants wrote somewhat longer messages in COMPOSE, 24.7 characters versus 20.2 in COPY.

In COMPOSE, participants created plausible and creative compositions. Table 4 shows some participant compositions after recognition by the decoder. To get an approximate error rate for COMPOSE, we used the crowdsourced judging procedure from [33]. We outline the procedure here for completeness.

	Entry rate (wpm)	Backspaces per character
COPY	33.4 ± 4.8 [13.9, 63.3]	0.048 ± 0.040 [0.00, 0.37]
COMPOSE	30.0 ± 4.4 [11.9, 51.0]	0.060 ± 0.046 [0.00, 0.42]
ANOVA test	$F_{1,23} = 10.5$, $\eta_p^2 = 0.31$, $p < .01$	$F_{1,23} = 4.6$, $\eta_p^2 = 0.17$, $p < .05$

	Task time (seconds)	Sentence input strategy (%)
COPY	14.5 ± 1.7 [9.8, 25.2]	36.5 ± 17.0 [0.0, 100.0]
COMPOSE	21.8 ± 3.4 [11.2, 40.2]	31.2 ± 17.1 [0.0, 100.0]
ANOVA test	$F_{1,23} = 24.5$, $\eta_p^2 = 0.52$, $p < .001$	$F_{1,23} = 1.6$, $\eta_p^2 = 0.07$, $p = 0.213$

Table 3. Results from Experiment 2. Participants copied sentences or composed their own messages. Results formatted as: mean ± 95% CI [min, max]. The bottom section of each table shows the repeated measures ANOVA statistical test for each dependent variable.

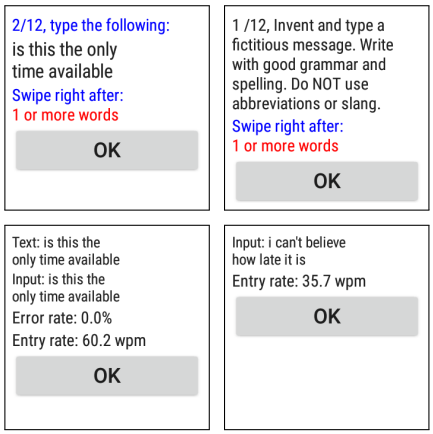

Figure 5. Prompts before each task (top) and post-input feedback (bottom) for Experiment 2: COPY (left) and COMPOSE (right).

you forgot your backpack
what do you want to do tonight
would you like to go hiking
ok good game see you tommoroww
seriously why does everyone hate pineapple on pizza
the weather kind of sus ks today

Table 4. Example text written by participants in Experiment 2 COMPOSE. This text includes any recognition errors made by the decoder.

We had nine workers on Amazon Mechanical Turk judge participants' compositions (including any recognition errors). Workers judged a set of compositions as correct, needing correction, or completely uncorrectable. We injected a set of easily correctable compositions, eliminating workers who failed to judge 70% of these correctly. After eliminating workers, all compositions were judged by at least six workers. For correctable cases, workers provided their best guess of the intended text, correcting obvious mistakes in spelling, grammar, punctuation, and case. If completely correct, a worker's *judged* CER was taken to be 0%. If completely incorrect, it was taken to be 100%. Otherwise we computed the CER ignoring case and punctuation other than apostrophes. We took the median CER of all workers. This resulted in a judged CER in COMPOSE of 2.2% ± 0.98 (95% CI). This was comparable to the CER in COPY of 2.6% ± 1.2 (95% CI).

We wanted to check how well our crowdsourced judging procedure from [33] measured the actual error rate. We did this by having nine workers judge each recognition result from the COPY condition where the reference text is actually available.

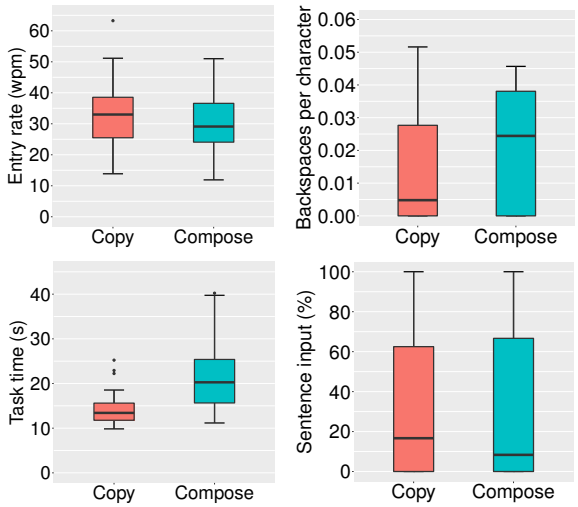

Figure 6. Entry rate, backspaces per character, task time, and sentence input strategy percent in Experiment 2.

The judged CER on COPY was 1.1% ± 0.54 (95% CI) compared to a CER of 2.6% ± 1.2 (95% CI) calculated using the actual reference text. The judged CER did seem to somewhat underestimate the CER. We conjecture this is due to workers either missing recognition errors altogether or correcting errors to similarly spelled words that may not have been the actual target word. While judged CER provides an approximate error rate for composition tasks, we recommend caution in interpreting its absolute magnitude.

As might be expected, participants were more tentative when composing. Participants erased characters more often: 0.060 backspaces per character in COMPOSE versus 0.048 in COPY. This difference was statistically significant (Table 3). Further, it took participants longer to complete each task. Including time spent on the prompt and post-input screens, participants took 21.8 s per task in COMPOSE versus only 14.5 s per task in COPY. This difference was statistically significant (Table 3).

This shows another tradeoff of a composition task. On the plus side, user behavior was perhaps closer to real-world text input, e.g. users changed their mind and erased previous text, users paused mid-sentence to think about what to write, etc. On the minus side, in a given time we can collect fewer tasks for estimating a participant's performance. Furthermore, a participant's performance may exhibit more variability due to cognitive overheads associated with creating novel text.

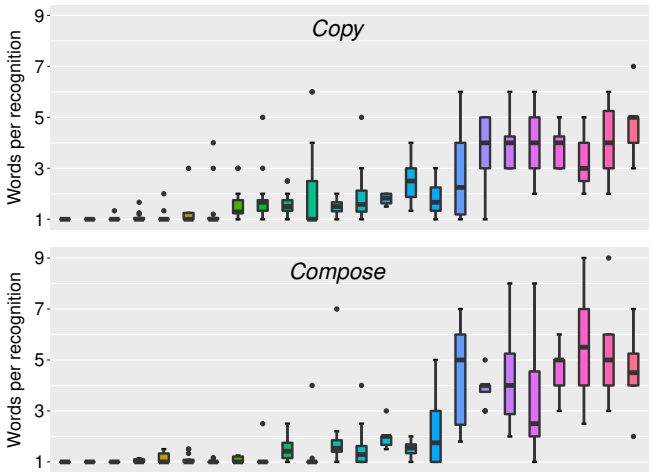

Figure 7. Words per recognition for each participant in COPY (top) and COMPOSE (bottom) in Experiment 2. Participants are ordered by mean words per recognition across both conditions. The same participant has the same x-position in both plots. The higher the words per recognition, the more that participant preferred multiple word or sentence input.

We measured whether participants used sentence input by counting how often tasks had only a single recognition event. We found sentence input was more frequent in COPY at 37% compared to COMPOSE at 31%. This difference was not statistically significant (Table 3).

Determining precisely whether a participant was using word or multiple-word input is more difficult. Participants may have switched input amounts during an entry. Furthermore, the decoder may return a different number of recognized words compared to a participant's intent. To provide a visualization of input amount, we plotted the number of words per recognition event in both conditions for each participant (Figure 7). A value of one indicates a word input strategy. As participants wrote on average four words in COPY and five in COMPOSE, values of four or more is indicative of sentence input. Values such as two are indicative of a multiple word input. Based on Figure 7, most participants seem to have adopted either a word or sentence input approach. Further, they seem to have used a similar approach in both conditions.

We also analyzed participants' input behavior by calculating the *difference* in words per recognition between conditions for each participant (subtracting a participant's COPY average from their COMPOSE average). 13 out of 24 participants had longer input in COMPOSE versus COPY. The magnitude of the difference was small: 0.13 ± 0.32 (95% CI). Thus it appears the entry task (copying text or composing novel text) did not have a large impact on participants' input size behavior.

We were curious if the width of a participant's index finger impacted performance. We split participants in two halves according to the median finger width. Figure 8 shows the performance of each participant in the text copy conditions of Experiment 1 and 2. The entry rate in the text copy conditions was 30.11 wpm $\pm$ 3.98 (95% CI) for the small finger group versus 30.05 wpm $\pm$ 7.93 (95% CI) for the large finger group. This difference was not statistically significant (Welsch t-test, $t_{16.2} = -0.02$, $p = 0.99$, $r = 0.004$). The error rate in the text

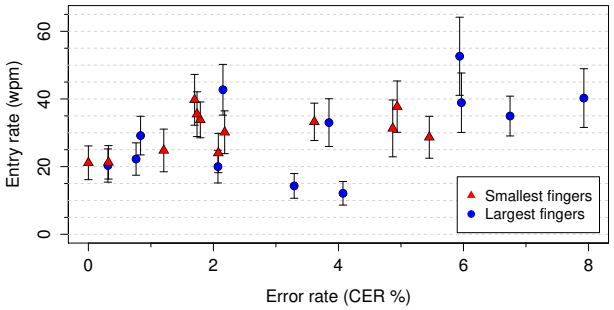

Figure 8. Entry and error rate of participants combining their data from Experiment 1 and COPY in Experiment 2. Vertical whiskers are one standard deviation. Red triangles are the 12 participants with smallest fingers, blue circles are the 12 participants with largest fingers.

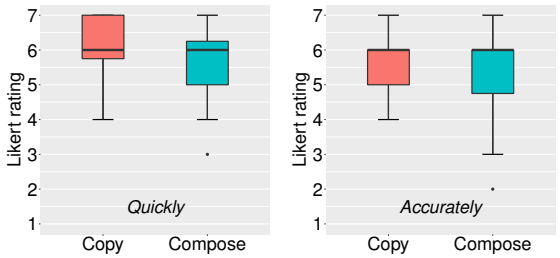

Figure 9. Subjective ratings about whether participants felt they entered text quickly (left) or accurately (right) in Experiment 2.

copy conditions was $2.49\% \pm 1.15$ (95% CI) for the small finger group versus $3.66\% \pm 1.61$ (95% CI) for the large finger group. This difference was not statistically significant (Welsch t-test, $t_{19.9} = 1.30$, $p = 0.21$, $r = 0.28$).

After each condition, participants rated statements on a 7-point Likert scale (1=strongly disagree, 7=strongly agree). The mean rating for the statement "I thought I entered text *quickly*" was 6.04 in COPY and 5.71 in COMPOSE (Figure 9 left). This difference was statistically significant (Friedman's test, $\chi^2(1) = 4.45$, $p < 0.05$). The mean rating for the statement "I thought I entered text *accurately*" was 5.79 in COPY and 5.42 in COMPOSE (Figure 9 right). This difference was not significant (Friedman's test, $\chi^2(1) = 0.82$, $p = 0.37$).

Open comments in Experiment 2 mostly related to recognition accuracy. However a number of participants commented on the difficulty of inventing compositions: "I began to run out of different sentences to write", "Typing my own message, I thought it was slower because I had to think about what to type, it felt more accurate though", and "My biggest downfall in this portion was thinking of words to say".

COMPUTATIONAL EXPERIMENTS

In this section, we conduct offline experiments using the data from our user study. Our goal is to inform deployment decisions for virtual keyboard decoders and to investigate whether we should be encouraging users to adopt larger input amounts.

Language Model and Search Pruning

Recall we had to aggressively prune our language models to fit within the memory constraints of a smartwatch. We compared the performance of the language models used in our

LM	Pruning	N-grams	Size	Perplexity
Char	SMALL	0.6 M	5.3 MB	3.83
	LARGE	5.0 M	48.5 MB	3.31
	UNPRUNED	757.8 M	7479.1 MB	3.05
Word	SMALL	0.8 M	6.3 MB	206.30
	LARGE	2.7 M	23.6 MB	166.34
	UNPRUNED	940.7 M	9986.7 MB	125.23

Table 5. Perplexity of different character and word language models.

study (SMALL) with less aggressively pruned models (LARGE) and with the original models before pruning (UNPRUNED). The LARGE models are of a size feasible for use on a more powerful mobile device (e.g. a phone) while the UNPRUNED models would require a cloud server.

We measured each language model's *perplexity* on the `mobile_nvp_nonum` sentences from the Enron mobile data set [31]. This set had 1,347 sentences containing lowercase letters, space, and apostrophe. Perplexity measures the average number of choices for the next character or word given the previous text. Lower perplexity is better.

As shown in Table 5, more aggressive pruning increased the perplexity of both the character and word language models. While decoding on-device removes the need for a network connected device, recognition accuracy may suffer compared to using a large language model hosted on a cloud server.

To verify these perplexity differences resulted in actual differences in recognition accuracy, we conducted experiments on the tap data recorded in Experiment 1. We additionally explored the impact of having more processing power. For these experiments, we used a 12-core 2.67 GHz server with the decoder configured to use 12 threads.

Our decoder's tradeoff between speed and accuracy is controlled by a search beam. For each of our language models sets, we tested three beams, the beam used in the study (NARROW) and two wider beams (MODERATE and WIDE). MODERATE represents a beam that is feasible on a more powerful mobile device while WIDE would likely require a cloud server.

Table 6 shows the mean participant CER for each language model set and beam combination on Experiment 1's tap data. For all the language model sets, using a MODERATE beam lowered error rate in all conditions. However this beam resulted in recognition taking on average eight times longer. Increasing to the WIDE beam did not provide any further error reduction.

The LARGE language models helped reduce recognition errors. However using the much larger UNPRUNED models did not provide further error reduction. In summary, the LARGE models with the MODERATE search beam was a good combination, resulting in substantially lower error rates, in particular for the sentence input which had its error rate cut in half. Further it seems that word-at-a-time entry is viable with modest memory and computation resources while multiple-word or sentence input is best done if more substantial resources are available.

Automatic Apostrophes

Participants frequently commented we failed to automatically insert apostrophes for input such as "im". Inserting omitted apostrophes is a common feature on commercial keyboards which often lack an apostrophe on the main keyboard. While our decoder can insert all valid characters during its search, it may have failed to find these probable apostrophe insertions due to our narrow search beam. We implemented a feature to allow apostrophes to be inserted with a separate penalty.

We randomly selected eight participants, tuning the apostrophe penalty on their data from Experiment 1 (all conditions) and Experiment 2 (COPY condition), 384 sentences. We tested the enhanced decoder on the remaining 16 participants, 768 sentences. Using the same language models and beam as the watch, the apostrophe insertion feature lowered error rate on from 2.85% to 2.49% CER. However, the more liberal insertion of apostrophes increased decoding time by 23%.

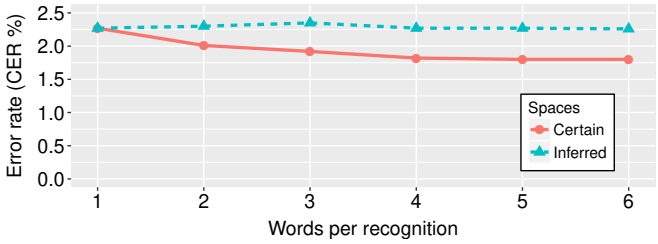

Figure 10. Error rate with increasing words per recognition. Spaces between words were either certain or needed to be inferred by the decoder.

Input Amount

An interesting question is whether providing more than one word improves recognition accuracy. Providing more words should allow the recognizer to make a more accurate guess since it can jointly infer a sequence of words. We tested this on 375 copy tasks from Experiment 1 and 2 in which participants exhibited a word-at-a-time input approach. We varied the number of words of tap data the recognizer was given before performing a decode. We also varied whether the decoder inserted a certain space character between words of tap data or if the taps were simply concatenated together. The earlier case simulates a user providing an input event that unambiguously delimits words (e.g. a right swipe). The later case simulates a user who relies on the decoder to insert all spaces. We used the LARGE language models and the MODERATE beam.

As shown in Figure 10, if spaces were known with certainty, error rate was reduced from 2.3% to 1.8%. This shows that the decoder could leverage the larger input sizes to make better guesses. However, if the spaces had to be inferred, the error rate stayed relatively constant at 2.3%. It seems the advantage of providing additional information during the decode was canceled out by the difficulty of guessing word boundaries.

The current design of multiple word input by many keyboards involves spaceless entry since the space key is used to trigger recognition. Our results suggest this may be suboptimal. By having separate input actions for indicating space and requesting recognition, recognition errors can be reduced. However this could slow input due to the additional space actions. Further work is needed to ascertain whether a design based on two input actions would yield improved end-user performance.

Language model	Search beam	Error rate (CER %)				Recognition time
		WORD	TWOWORD	SENTENCE	Average	(ms)
SMALL	NARROW	3.09	2.57	3.94	3.20	3.2
	MODERATE	2.96	2.38	2.31	2.55	25.8
	WIDE	2.96	2.39	2.31	2.55	301.3
LARGE	NARROW	2.91	2.86	2.80	2.86	3.5
	MODERATE	2.78	2.49	2.04	2.44	25.4
	WIDE	2.78	2.49	2.01	2.43	273.3
UNPRUNED	NARROW	2.85	2.35	2.89	2.70	4.6
	MODERATE	2.76	2.30	2.24	2.43	29.9
	WIDE	2.76	2.22	2.10	2.36	482.7

Table 6. Error rates and recognition time on data from Experiment 1 varying the language model size and search beam. Results on a 12-core 2.67 GHz server. Recognition time is the average over all three conditions. The first row is the smartwatch configuration used in the user study.

DISCUSSION

As expected, requiring fewer motor actions resulted in increased entry speed. Even in the challenging environment of input on a tiny touchscreen device with severe resource constraints, we found we could accurately recognize mobile messages with error rates below 4%. We found users easily adopted various input size strategies. When allowed to choose their own input size strategy, users were split between conventional word-at-a-time approach and longer phrase- or sentence-at-a-time strategies. We found this was true both when they were copying provided sentences and when they composed their own sentences.

While we have shown it is possible to increase input speeds and lower recognition error rates via larger input amounts, this does not necessarily mean a user's *effective* entry and error rates are guaranteed to improve. Recognition errors may occur and need correction. A key challenge is how to design error correction interfaces that encourage users to "go big" with the confidence that any recognition errors can be corrected quickly and easily. If correction interfaces are not well done, any gains from providing larger amounts of input can easily be lost in time consuming and frustrating correction episodes.

Our keyboard required interface actions to trigger recognition, delete taps, or move to the next task. Adding buttons big enough to be reliably hit would have required a large amount of screen space. Instead, we opted to use swipe gestures. On the plus side, most users easily learned to use our swipe gestures. On the minus side, some users felt swiping was slow compared to tapping. Indeed, right swipes were almost twice as long as discrete taps in Experiment 1 (112 ms versus 68 ms). It would be interesting to explore other options, e.g. using the text result area as a big button or mapping multi-finger taps to actions. Designing a small set of highly reliable input actions is an important consideration for interfaces that rely on uncertain input as they provide anchor points where the user and system can synchronize after a misunderstanding.

We found error rates were low even when using highly pruned language models. However, the Enron memorable sentences tend to be rather easy to predict English. It would be interesting to see if larger language models are required for accurate recognition of more diverse text, e.g. text with uncommon words, niche topics, or text about current events.

Our results confirm that the theoretical motor time reduction afforded by larger input amounts does indeed result in faster entry rates. However, whether larger input amounts reduce error rates is unclear. While in our study the error rate after recognition was higher for sentence input, post-hoc experiments revealed this was due to over-pruning in the decoder's search. With sufficient computation, it would be possible to achieve a lower error rate for sentence input. Moreover, while we originally thought the advantage of larger input would come from providing more information at one time to the decoder, our experiments showed that accuracy was constant across different input amounts if spaces had to be inferred.

Overall, sentence-based input does not appear to necessarily reduce error rate from a decoding-perspective. One possible explanation is that the additional error rate reduction due to additional information about the user's intention is to some extent canceled out by the lack of any user input that specifies spaces. However, it should be emphasized that keyboards could allow users to signal spaces in some way. Moreover, we found that users tended to be more precise as their input amount increases, possibly due to the less context switching. While we have shown larger input amounts increase entry rates, it remains to be seen whether lower errors rates can also be provided in practice for virtual keyboard input.

CONCLUSIONS

We explored how to improve text entry speed and recognition accuracy when using a virtual keyboard decoder. We focused on allowing users to change the quantity of observations provided to the decoder for each recognition event.

Compared to a word-at-a-time input strategy, providing input of an entire sentence significantly improved entry rates from 26 wpm to 32 wpm. This was done while keeping the character error rate below 4%. In offline recognition experiments, we found that with more processing power and memory, sentence input could be recognized with a 2.0% error rate while word-at-a-time input had a higher 2.8% error rate.

Our results suggest recognition-based touchscreen input methods can be designed to enhance performance by allowing users to modulate the amount of input per recognition event.

Acknowledgements

This work was supported by Google Faculty awards (K.V. and P.O.K.), and EPSRC EP/N010558/1, EP/N014278/1 (P.O.K.).

REFERENCES

1. Arif, A. S., and Mazalek, A. A survey of text entry techniques for smartwatches. In *Proceedings, Part II, of the 18th International Conference on Human-Computer Interaction. Interaction Platforms and Techniques - Volume 9732*, Springer-Verlag New York, Inc. (New York, NY, USA, 2016), 255–267.
2. Bi, X., Ouyang, T., and Zhai, S. Both complete and correct?: Multi-objective optimization of touchscreen keyboard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, ACM (New York, NY, USA, 2014), 2297–2306.
3. Bi, X., Smith, B. A., and Zhai, S. Quasi-qwerty soft keyboard optimization. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2010), 283–286.
4. Bi, X., Smith, B. A., and Zhai, S. Multilingual touchscreen keyboard design and optimization. *Human–Computer Interaction 27*, 4 (2012), 352–382.
5. Chelba, C., Brants, T., Neveitt, W., and Xu, P. Study on interaction between entropy pruning and kneser-ney smoothing. In *Proceedings of Interspeech* (2010), 2242–2245.
6. Garay-Vitoria, N., and Abascal, J. Text prediction systems: a survey. *Universal Access in the Information Society 4*, 3 (2006), 188–203.
7. Goodman, J., Venolia, G., Steury, K., and Parker, C. Language modeling for soft keyboards. In *Eighteenth National Conference on Artificial Intelligence*, AAAI '02, American Association for Artificial Intelligence (Menlo Park, CA, USA, 2002), 419–424.
8. Gordon, M., Ouyang, T., and Zhai, S. Watchwriter: Tap and gesture typing on a smartwatch miniature keyboard with statistical decoding. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '16, ACM (New York, NY, USA, 2016), 3817–3821.
9. Jurafsky, D., and Martin, J. H. *Speech and language processing*. Pearson London, 2000.
10. Kristensson, P. O. *Discrete and Continuous Shape Writing for Text Entry and Control*. PhD thesis, Linköping University, 2007.
11. Kristensson, P. O. Five challenges for intelligent text entry methods. *AI Magazine 30*, 4 (2009), 85–94.
12. Kristensson, P. O. Next-generation text entry. *Computer 48*, 7 (2015), 84–87.
13. Kristensson, P. O., and Vertanen, K. Performance comparisons of phrase sets and presentation styles for text entry evaluations. In *Proceedings of the 2012 ACM International Conference on Intelligent User Interfaces*, IUI '12, ACM (New York, NY, USA, 2012), 29–32.
14. Kristensson, P. O., and Vertanen, K. 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 and Services*, MobileHCI '14, ACM (New York, NY, USA, 2014), 335–338.
15. Kristensson, P.-O., and Zhai, S. Shark^2: A large vocabulary shorthand writing system for pen-based computers. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology*, UIST '04, ACM (New York, NY, USA, 2004), 43–52.
16. Kristensson, P. O., and Zhai, S. Relaxing stylus typing precision by geometric pattern matching. In *IUI '05: Proceedings of the 10th International Conference on Intelligent User Interfaces*, ACM Press (2005), 151–158.
17. Leiva, L. A., Sahami, A., Catala, A., Henze, N., and Schmidt, A. Text entry on tiny qwerty soft keyboards. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, ACM (New York, NY, USA, 2015), 669–678.
18. MacKenzie, I. S., and Soukoreff, R. W. Text entry for mobile computing: Models and methods, theory and practice. *Human–Computer Interaction 17*, 2-3 (2002), 147–198.
19. MacKenzie, I. S., and Soukoreff, R. W. 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, 2003), 754–755.
20. MacKenzie, I. S., and Tanaka-Ishii, K. *Text Entry Systems*. Morgan Kauffman, 2007.
21. MacKenzie, I. S., and Zhang, S. X. The design and evaluation of a high-performance soft keyboard. In *CHI '99: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM Press (New York, NY, USA, 1999), 25–31.
22. Oulasvirta, A., Reichel, A., Li, W., Zhang, Y., Bachynskyi, M., Vertanen, K., and Kristensson, P. O. Improving two-thumb text entry on touchscreen devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2013), 2765–2774.
23. Rabiner, L., and Juang, B.-H. *Fundamentals of Speech Recognition*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.
24. Rick, J. Performance optimizations of virtual keyboards for stroke-based text entry on a touch-based tabletop. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, ACM (2010), 77–86.
25. Smith, B. A., Bi, X., and Zhai, S. Optimizing touchscreen keyboards for gesture typing. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, ACM (2015), 3365–3374.
26. Stolcke, A. Entropy-based pruning of backoff language models. In *Proceedings of DARPA Broadcast News Transcription and Understanding Workshop* (1998), 270–274.

27. Turner, C. J., Chaparro, B. S., and He, J. Text input on a smartwatch qwerty keyboard: Tap vs. trace. *International Journal of Human-Computer Interaction 33*, 2 (2017), 143–150.

28. Vertanen, K., and Kristensson, P. O. Automatic selection of recognition errors by respeaking the intended text. In *ASRU '09: IEEE Workshop on Automatic Speech Recognition and Understanding* (December 2009), 130–135.

29. Vertanen, K., and Kristensson, P. O. Parakeet: A continuous speech recognition system for mobile touch-screen devices. In *Proceedings of the 14th International Conference on Intelligent User Interfaces*, IUI '09, ACM (New York, NY, USA, 2009), 237–246.

30. Vertanen, K., and Kristensson, P. O. Getting it right the second time: Recognition of spoken corrections. In *SLT '10: Proceedings of the 3rd IEEE Workshop on Spoken Language Technology* (December 2010), 277–282.

31. Vertanen, K., and Kristensson, P. O. 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*, MobileHCI '11, ACM (New York, NY, USA, 2011), 295–298.

32. Vertanen, K., and Kristensson, P. O. Complementing text entry evaluations with a composition task. *ACM Transactions on Computer-Human Interaction 21*, 2 (2014), 8:1–8:33.

33. Vertanen, K., and Kristensson, P. O. Complementing text entry evaluations with a composition task. *ACM Transactions on Computer-Human Interactions 21*, 2 (Feb. 2014), 8:1–8:33.

34. Vertanen, K., Memmi, H., Emge, J., Reyal, S., and Kristensson, P. O. 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*, CHI '15, ACM (New York, NY, USA, 2015), 659–668.

35. Weir, D., Pohl, H., Rogers, S., Vertanen, K., and Kristensson, P. O. Uncertain text entry on mobile devices. In *Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems*, CHI '14, ACM (New York, NY, USA, 2014), 2307–2316.

36. Yi, X., Yu, C., Shi, W., Bi, X., and Shi, Y. Word clarity as a metric in sampling keyboard test sets. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, ACM (New York, NY, USA, 2017), 4216–4228.

37. Yi, X., Yu, C., Shi, W., and Shi, Y. Is it too small?: Investigating the performances and preferences of users when typing on tiny qwerty keyboards. *International Journal of Human-Computer Studies 106*, Supplement C (2017), 44–62.

38. Young, S. J., Russell, N., and Thornton, J. Token passing: A simple conceptual model for connected speech recognition systems. Tech. rep., Cambridge University Engineering Department, 1989.

39. Zhai, S., Hunter, M., and Smith, B. A. The metropolis keyboard-an exploration of quantitative techniques for virtual keyboard design. In *Proceedings of the 13th annual ACM symposium on User interface software and technology*, ACM (2000), 119–128.

40. Zhai, S., Hunter, M., and Smith, B. A. Performance optimization of virtual keyboards. *Human–Computer Interaction 17*, 2-3 (2002), 229–269.

41. Zhai, S., and Kristensson, P.-O. Shorthand writing on stylus keyboard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03, ACM (New York, NY, USA, 2003), 97–104.

42. Zhai, S., and Kristensson, P. O. Interlaced qwerty: accommodating ease of visual search and input flexibility in shape writing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2008), 593–596.

43. Zhai, S., and Kristensson, P. O. The word-gesture keyboard: Reimagining keyboard interaction. *Communications of the ACM 55*, 9 (2012), 91–101.

44. Zhai, S., Kristensson, P.-O., and Smith, B. A. In search of effective text input interfaces for off the desktop computing. *Interacting with computers 17*, 3 (2004), 229–250.

45. Zhai, S., Sue, A., and Accot, J. Movement model, hits distribution and learning in virtual keyboarding. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM (2002), 17–24.