ContextType: Using Hand Posture Information to Improve Mobile Touch Screen Text Entry

Mayank Goel[1], Alex Jansen[2], Travis Mandel[1], Shwetak N. Patel[1], Jacob O. Wobbrock[2]

[1]Computer Science & Engineering | DUB Group
University of Washington
Seattle, WA 98195 USA
{mayank, tmandel, shwetak}@cs.washington.edu

[2]The Information School | DUB Group
University of Washington
Seattle, WA 98195 USA
{ajansen7, wobbrock}@uw.edu

ABSTRACT

The challenge of mobile text entry is exacerbated as mobile devices are used in a number of situations and with a number of hand postures. We introduce *ContextType*, an adaptive text entry system that leverages information about a user's hand posture (using two thumbs, the left thumb, the right thumb, or the index finger) to improve mobile touch screen text entry. ContextType switches between various keyboard models based on hand posture inference while typing. ContextType combines the user's posture-specific touch pattern information with a language model to classify the user's touch events as pressed keys. To create our models, we collected usage patterns from 16 participants in each of the four postures. In a subsequent study with the same 16 participants comparing ContextType to a control condition, ContextType reduced total text entry error rate by 20.6%.

Author Keywords

Touch screen; situational impairments; mobile devices; hand posture; grip; text entry; virtual keyboard.

ACM Classification Keywords

H.5.2. Information interfaces and presentation: User Interfaces – *input devices and strategies.*

INTRODUCTION

In comparison to traditional desktop keyboards, text entry on touchscreen mobile devices is more challenging. These devices are as small as the palm of a hand and are used in a number of dynamic environments. These factors can lead to situational impairments [8], which can pose significant challenges to effective interaction because our current mobile devices do not have much awareness of our environments, and thus cannot adapt to them.

Mobile devices may be used in a number of different hand postures. Azenkot and Zhai [1] found that majority of users at least "sometimes" used their phones with either the thumb of their dominant hand, their dominant index finger, or both thumbs. In addition, due to variety of contexts in which mobile devices are used, there are situations where the user's dominant hand is occupied (*e.g.*, supporting oneself with a grab-handle while standing on a moving bus) and the device is operated with the non-dominant hand's thumb. Research has shown that mobile device hand postures can significantly affect finger and thumb pointing performance [11], but such information has not been used for improving text entry, which requires numerous rapid, accurate strikes and is a relatively high-intensity, if familiar, task.

In this paper, we present *ContextType,* a system that infers users' hand postures to improve text entry on mobile touch screen devices. ContextType supports typing with four hand postures: two thumbs, just the left thumb, just the right thumb, and either index finger. ContextType switches between underlying touch-models based on inference about how the user is holding the device while typing, without changing the visual layout of the keyboard. ContextType leverages our previous work on *GripSense* [4], which infers hand posture as left thumb, right thumb, or index finger without any additional sensors. ContextType also detects two-thumbed hand postures without adding any external sensors to the device. Once posture is inferred, ContextType combines a user's posture-specific touch-pattern information with a language model to classify the user's touch event as a pressed key, ultimately making text entry more accurate.

To design and build ContextType, we first collected touch screen typing data from 16 participants in all four hand-postures. Based on this data we built touch-based key-press classification models, one for each hand posture, personalized for each participant. The final ContextType prototype is a composite of these personalized touch models and a 5-gram language model. These models will be discussed in more detail below.

We evaluated the final ContextType system, with and without the language model, in a study with the same 16 participants. The control keyboard, to which ContextType was compared, also used personalized keyboards for each participant, but did not take hand posture into account. Our findings show that ContextType decreases total text entry error rate by 20.6%. We also found that inclusion of a language model does not have a significant improvement over the control condition.

The main contribution of this paper is a demonstration that knowledge of a user's hand posture *can* be used to improve

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
CHI 2013, April 27–May 2, 2013, Paris, France.
Copyright © 2013 ACM 978-1-4503-1899-0/13/04...$15.00.

typing performance on mobile devices. This contribution comes in two parts: (1) ContextType itself, which detects a user's hand posture in real time and selects a personalized, posture-specific keyboard model; and (2) empirical evidence for the benefit of adding hand posture information to a personalized keyboard.

DESIGN OF CONTEXTTYPE

ContextType combines three types of information to classify users' touch events as key presses. It is informed by data about a user's hand posture, by a user's touch pattern, and also by letter probabilities from a language model. The algorithm develops different models for different postures. The inference of the hand posture uses techniques from GripSense [4], which can infer the left thumb, right thumb, or either index finger. ContextType extends this functionality by also inferring two-thumbed postures.

Two-Thumbed Posture Detection

The functionality to detect a two-thumbed posture uses tap sizes and time elapsed between taps. GripSense differentiates between left and right thumb usage by observing tap sizes. Tap sizes increase as the user touches the far side of the screen, *i.e.*, when operated with the left thumb, the areas touched on the right side of the screen will be bigger than those on the left side and vice versa. ContextType observes this phenomenon and applies it to two thumbs, inferring a two-handed two-thumb posture if the tap sizes in the center of the screen are 25% larger than those on either side. Another heuristic that is combined with tap sizes is the relative difference in time elapsed between taps on either side of the screen. When using one hand to operate the phone, it takes longer to go from one side of the screen to the other. Hence, if a tap on one side of the screen is followed by another tap on the opposite side, the time difference will be larger than the average time difference because the thumb/finger needs to travel farther. In contrast, when operated with two thumbs, the time difference between taps on opposite sides of the screen will be significantly less. Hence, the system inferred two-thumbed interaction if the difference in time interval between taps on opposite sides and the mean time interval between taps was greater than 30%.

The implementation details for the detection of other hand postures can be found in our prior work on GripSense [4]. Our offline analysis showed that ContextType was able to detect hand posture with an average accuracy of 89.7% and the decision was made within 4.9 taps, on average.

Touch Pattern Model

ContextType personalizes the underlying keyboard layout by modifying the motor-space location of the keys according to the user's typing behavior (*i.e.,* the visual layout of the keyboard remains static). ContextType employs a constant, diagonal covariance structure by computing a bivariate Gaussian distribution [3] for each key and centers each key at the centroids of predicted key-presses that are personalized for each user. Considering that touch behavior varies not only across participants but also across hand postures for the same participant [1], we generate separate models for different hand postures for each participant. Figure 1 shows a sample of the variance in touch behavior for a participant. In the case of single thumbs, it is clear that the user tended to bias towards the side of the thumb because of its limited reach.

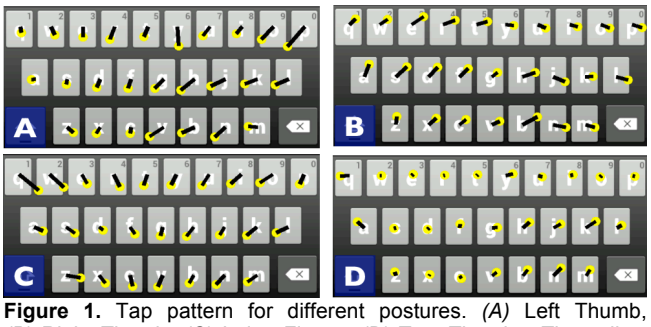

Figure 1. Tap pattern for different postures. *(A)* Left Thumb, *(B)* Right Thumb, *(C)* Index Finger, *(D)* Two Thumbs. The yellow spot is the touch centroid for each key and the line shows drift from the visual key center.

Language Model

We implemented a 5-gram letter model following the work of [5]. We trained the model on the Brown corpus [6], consisting of American English from a variety of sources. We employ the Modified Kneser-Ney method for probability smoothening, which has been successfully used by Chen *et al.* [2] in language modeling for soft keyboards. The validity and effectiveness of the language model was confirmed in a small study (6 participants), similar in apparatus and design to the ContextType evaluation described in next section. In this study, the participants were only required to complete 40 phrases in any one of their preferred hand postures. This study compared a static, non-adaptive keyboard to a language model-powered keyboard. There was a significant increase in words per minute for the language model ($F_{1,315}$=57.14, p<.0001). The language model also resulted in decreased error rates (Wilcoxon signed-rank test: Z=-2717.5, p<.0001).

Combining Touch and Language Models

ContextType combines touch and language models by calculating probabilities for each key. The most likely intended key, k_i^*, is given by:

$$k_i^* = \text{argmax}_{k_i} p_L(k_i|h).p_T(k_i|l)$$

where p_L is the language model probability, p_T is the touch model probability, k_i is the probability for each key, h is the language model history (last 4 entered characters in case of 5-gram letter model), and $l \in R^2$ is an x and y coordinate pair denoting the last touch location on the screen

Data Collection

The touch models were built based on typing data collected from 16 participants (9 males, 7 females) who each volunteered for a 45-minute study session. All participants self-rated as expert computer users and intermediate to

expert touch screen mobile device users. Participants were between 22 and 33 years of age (M = 27.38, SD = 2.7).

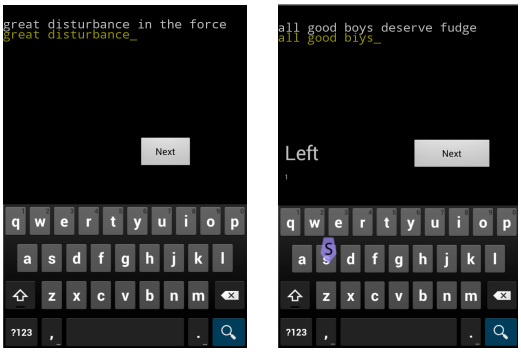

Figure 2. *(Left)* Data collection interface. *(Right)* Test interface with current posture and keystroke feedback.

We built a custom data collection application for the Android OS (Figure 2) using a Samsung Galaxy Nexus phone. The interface was designed in a way to capture the user's natural typing pattern. Thus, it did not inform users of their mistakes and the correct letter was always displayed for each key press. The interface also allowed a swipe from right-to-left to remove the last character typed. Participants were instructed to swipe when they felt that they had made an error. Noisy data was removed by filtering out taps that landed outside the Euclidean bounds of the intended key or its immediate neighbors. Once comfortable with the interface, the participants were asked to enter 30 phrases in each of the 4 hand postures. The order of postures was counterbalanced. Short phrases of English text from MacKenzie and Soukoreff's phrase set were used [7]. Apart from these phrases, every fifth phrase was a randomly selected pangram from a list of 35 pangrams to ensure sufficient representation of all letters of English alphabet.

EVALUATION

We sought to see whether the knowledge of a user's hand posture could be used to improve text entry performance. In addition, we also wanted to investigate the effect of the language model on the overall performance of ContextType.

Participants. The same 16 participants who participated in the data collection phase were used for a second session, lasting approximately 1 hour, to evaluate ContextType.

Apparatus. Participants used a similar interface to the one used during the data collection phase. This time, the entered text contained the actual key classification and visual keystroke feedback (Figure 2, *left*).

Procedure. The session began with an introduction to the modified interface and explanation of the task. For each condition, participants completed 40 phrases. The application instructed the user to change hand posture after every five phrases. In the bottom-left corner of the text area, the current phrase number and current expected hand posture were displayed. The hand postures were counterbalanced and selected randomly (Figure 2, *right*).

Design & Analysis. The study was a within-subjects 2×2×4 factorial design. The factors and levels were:

- ***Interface:*** *ContextType*, *Control*.
- ***Language Model:*** *Yes*, *No*.
- ***Posture:*** *Left Thumb, Right Thumb, Index Finger, Two-Thumbs*.

When ContextType was running, the keyboard was personalized by leveraging touch data collected for each user and each of his or her hand postures. In the *Control* condition, the touch data was not partitioned for each hand posture. Hence, the control condition, though not adaptive to hand posture, had a personalized keyboard. Presentation of conditions was counterbalanced. With 40 phrases in each condition, participants entered 2×2×40 = 160 phrases each.

The main measures were speed, calculated as words per minute (WPM), and total error rate [9]. Total error rate is decomposed into corrected and uncorrected error rates. Corrected errors are the errors that are subsequently corrected by the user before moving on to the next phrase. Uncorrected errors are those that are left in the transcribed phrase at the end of each trial. Also, the participants were asked to rate which of the conditions they preferred and why. The participants were not aware which condition was the current one to prevent bias.

For WPM, we present results from a mixed-effects model analysis of variance. For error rates, we used the nonparametric Aligned Rank Transform procedure [10]. We used a nonparametric analysis for error rates because error rates skew towards zero and violate normality. All pairwise comparisons were protected against Type I error using Holm's sequential Bonferroni procedure.

RESULTS

Speed. There was no detectable difference in speed owing to ContextType (27.5 WPM, SD=6.6 *vs.* 26.0 WPM, SD=6.2; $F_{1,2496}$=1.27, *n.s.*). However, *Posture* had a significant effect on WPM ($F_{1,2496}$=92.06, p<.0001). This was expected because participants generally preferred some posture to another. *Post hoc* pairwise comparisons showed that all postures were significantly different and that two thumbs were fastest, followed by right thumb, index finger, and left thumb. Left thumb's lower performance was expected because it was the non-dominant thumb for all participants. There was no significant *ContextType*×*Posture* interaction. Finally, there was no detectable increase in speed due to the language model (26.5 WPM, SD=6.3 *vs.* 27.0 WPM, SD=6.6; $F_{1,2496}$=2.20, *n.s.*).

Error Rate. Corrected error rates are subsumed in typing speed because correcting errors slows users down. Considering there was no detectable effect of ContextType on speed, we analyzed corrected error rates to investigate ContextType's performance further. Participants exhibited a marked and significant improvement in corrected error rate while using *ContextType* (4.86%, SD=2.4 *vs.* 6.49%, SD=4.4; $F_{1,2496}$=9.79, p<.002). *Language Model* also

resulted in a trend toward reduced corrected error rates ($F_{1,2496}$=3.09, p=.08).

There was no detectable difference in uncorrected error rate due to ContextType (2.38%, *SD*=1.58 *vs.* 2.63%, *SD*=2.35; $F_{1,2496}$=0.06, *n.s.*). No other factors had a significant effect on uncorrected error rate.

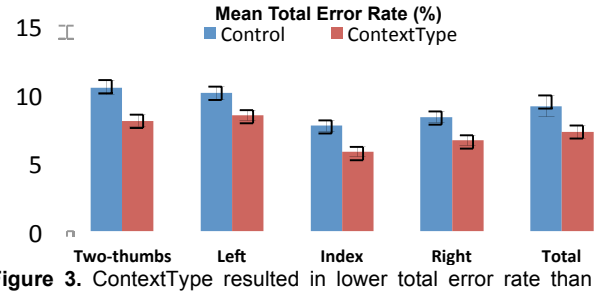

Figure 3. ContextType resulted in lower total error rate than the control condition for all the four hand postures. Error bars are standard errors.

We also evaluated ContextType's effect on total error rate, which is the sum of corrected and uncorrected error rates. We observed a significant effect of *ContextType* on total error rate ($F_{1,2496}$=10.87, p<.002). Compared to the control condition, total error rates decreased by 20.6% (Figure 3).

In contrast to corrected error rate, the *Language Model* did not significantly affect total error rate. However, there was a significant *ContextType×Language Model* interaction ($F_{1,2496}$=3.94, p<.05). However, no *post hoc* pairwise comparisons were significant. As we would expect, there was a significant effect of *Posture* on total error rate ($F_{3,2496}$=4.97, p<.002). *Post hoc* pairwise comparisons showed that the left thumb was significantly less accurate than both the index finger ($F_{1,2496}$=12.12, p<.001) and the right thumb ($F_{1,2496}$=6.31, p<.02). This result was expected because left thumb was the non-dominant thumb for all 16 participants. The performance with two thumbs was also found to be significantly less accurate than that of the index finger ($F_{1,2496}$=6.82, p<.01). Considering two thumbs had significantly higher WPM than the index finger, it suggests a speed and accuracy trade-off between the two postures.

Preference. We asked participants which of the two interfaces they preferred. Nine out of 16 participants chose ContextType. The remaining 7 participants did not perceive any performance difference. P7 said, "[ContextType] was awesome! I did not have to look at the keyboard and the 'P' key felt much closer and accessible with my left hand".

DISCUSSION

ContextType decreased corrected error rate significantly, but no significant effect on WPM was observed. Corrected error rate is generally correlated with WPM, which suggests that with more data, ContextType's improvement in typing speed might be detectable. Also, the decreased accuracy of ContextType for detecting posture (89.7%) might not be an impediment to its performance. Anecdotally, we analyzed results for one participant and found that ContextType primarily confused her index finger and right thumb. Upon further analysis we found that her typing pattern for index finger and right thumb were similar, thereby producing similar touch models. Our posture detection system does not require any calibration; hence posture-specific keyboard touch-models can be refined over continued usage without any user intervention. Although different language model implementations could produce significant improvements, and our language model *did* improve performance over a static keyboard, our language model did not result in an improvement in performance over a keyboard using personalized touch models. It seems that the language model's benefit is largely negated in the presence of a personalized touch model.

CONCLUSION

ContextType detects a user's hand posture (two thumbs, the left thumb, the right thumb, or the index finger) and combines posture-specific touch pattern information with a language model to inform an adaptive keyboard. We conducted a study to collect usage patterns from 16 participants in each of the four postures. In an evaluation with the same 16 participants, ContextType reduced total error rate by 20.6%. Hence, ContextType shows that making our mobile devices more aware of their users can improve both those devices and the experience of the users who use them.

ACKNOWLEDGEMENTS

This work was supported in part by the National Science Foundation under grant IIS-1217627.

REFERENCES

1. Azenkot, S. and Zhai, S. (2012). Touch behavior with different postures on soft smartphone keyboards. *Proc.. MobileHCI'12*. New York: ACM Press, pp. 251-260.
2. Chen, S.F. and Goodman, J. (1996). An empirical study of smoothing techniques for language modeling. *Proc. Assoc. Comp. Ling.,1996*. Stroudsberg, pp. 310-318.
3. Findlater, L. and Wobbrock, J. (2012). Personalized input: Improving ten-finger touchscreen typing through automatic adaptation. *Proc. CHI'12*. New York: ACM Press, pp. 815-824.
4. Goel, M., Wobbrock, J.O., and Patel, S.N. (2012). GripSense: Using built-in sensors to detect hand posture and pressure on commodity mobile phones. *Proc. UIST'12*. New York: ACM Press, pp. 545-554.
5. Goodman, J., Venolia, G., Steury, K., and Parker, C. (2002). Language modeling for soft keyboards. *Proc. IUI'02*. New York: ACM Press, pp. 194-195.
6. Kucera, H. and Francis, W. (1967). *Computational Analysis of Present-Day American English.*
7. MacKenzie, I.S. and Soukoreff, R.W. (2003). Phrase sets for evaluating text entry techniques. *Proc. CHI'03 EA*. ACM Press.
8. Sears, A., Lin, M., Jacko, J., and Xiao, Y. (2003). When computers fade pervasive computing and situationally-induced impairments and disabilities. *HCI International 2'03*, pp. 1298-1302.
9. Soukoreff, R.W. and MacKenzie, I.S. (2003). Metrics for text entry research: an evaluation of MSD and KSPC, and a new unified error metric. *Proc. CHI'03*. New York: ACM Press, pp. 113-120.
10. Wobbrock, J.O., Findlater, L., Gergle, D., and Higgins, J.J. (2011). The aligned rank transform for nonparametric factorial analyses using only anova procedures. *Proc. CHI'11*. New York: ACM Press, pp. 143-146.
11. Wobbrock, J.O., Myers, B.A., and Aung, H.H. (2008). The performance of hand postures in front- and back-of-device interaction for mobile computing. *International Journal of Human-Computer Studies 66,12*. Duluth, MN: Academic Press, pp. 857-875.