ON–THE–GO TEXT ENTRY: EVALUATING AND IMPROVING MOBILE TEXT INPUT ON MINI–QWERTY KEYBOARDS

A Thesis
Presented to
The Academic Faculty

by

James Clawson

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Interactive Computing

Georgia Institute of Technology
November 2012

ON–THE–GO TEXT ENTRY: EVALUATING AND IMPROVING MOBILE TEXT INPUT ON MINI–QWERTY KEYBOARDS

To my fiancée,

Dana Habeeb,

without your love and support, none of this would have been possible.

Thank you.

ACKNOWLEDGEMENTS

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

To date, hundreds of millions of mini-QWERTY keyboard equipped devices (miniaturized versions of a full desktop keyboard) have been sold. Accordingly, a large percentage of text messages originate from fixed-key, mini-QWERTY keyboard enabled mobile phones. Over a series of three longitudinal studies I quantify how quickly and accurately individuals can input text on mini-QWERTY keyboards. I evaluate performance in ideal laboratory conditions as well as in a variety of mobile contexts. My first study establishes baseline performance measures; my second study investigates the impact of limited visibility on text input performance; and my third study investigates the impact of mobility (sitting, standing, and walking) on text input performance. After approximately five hours of practice, participants achieved expertise typing almost 60 words-per-minute at almost 95% accuracy. Upon completion of these studies, I examine the types of errors that people make when typing on mini-QWERTY keyboards. Having discovered a common pattern in errors, I develop and refine an algorithm to automatically detect and correct errors in mini-QWERTY keyboard enabled text input. I both validate the algorithm through the analysis of pre-recorded typing data and then empirically evaluate the impacts of automatic error correction on live mini-QWERTY keyboard text input. Validating the algorithm over various datasets, I demonstrate the potential to correct approximately 25% of the total errors and correct up to 3% of the total keystrokes. Evaluating automatic error detection and correction on live typing results in successfully correcting 60.80% of the targeted errors committed by participants while increasing typing rates by almost two words-per-minute without introducing any distraction.

CHAPTER I

INTRODUCTION

The mobile phone is rapidly becoming the most widely adopted computing device used today. There are over 6 billion mobile phone subscribers in the world, with some researchers predicting that that the number of mobile subscriptions will grow to exceed the earth's population (7 billion) by the end of 2012 and will exceed 10 billion by 2016 [8]. Text messaging is an important facet of mobile phone usage. For example, over 2 trillion messages were sent in the United States in 2011 [47] which corresponds to 6 billion text messages sent per day. These statistics are remarkable considering the inefficiencies and poor design of current text entry methods for mobile devices.

Recently, mobile phone manufacturers have exhibited awareness of the need to build devices that enable more rapid text input. With the increasing popularity and adoption of mini–QWERTY keyboard enabled mobile phones, it is critical that we explore the human-computer interaction issues associated with these devices. In this dissertation, I detail my research on the evaluation of mini–QWERTY keyboards — miniature versions of the traditional desktop QWERTY keyboard built into many mobile devices as a means of text entry. Portions of this research have been previously published in venues such as ISWC [12], MobileHCI [13] and CHI[9, 10, 11].

I present my thesis statement below. After introducing the thesis, I enumerate my contributions and discuss how each contribution is supported by a chapter of the dissertation.

1.1 Thesis Statement

FatThumbs, a method for automatically detecting and correcting typographical errors associated with pressing multiple keys at once in mini-QWERTY keyboard mobile text input, improves the text entry experience by reducing errors without distracting the user.

1.2 Contributions

The exploration of this thesis yields the following contributions:

1. A longitudinal laboratory study to determine baseline typing performance on mini–QWERTY keyboards (Chapter 3).
2. A longitudinal laboratory study examining typing performance in conditions of limited visibility (Chapter 4).
3. A longitudinal study of expert typing while mobile (sitting, standing, and walking) (Chapter 5).
4. An analysis of typographical errors made on mini–QWERTY keyboards in various mobile contexts (Chapter 6).
5. An algorithm that automatically detects and correct errors in mini–QWERTY keyboard typing (Chapter 7 and 8).
6. A laboratory study evaluating the impacts of automatic error detection and correction on live expert typing performance (Chapter 9).

1.3 Overview of the Dissertation

In Chapter 2 I present an overview of text entry research relevant to this dissertation. I review research on physical keyboards, error detection methodologies and metrics, and use-in-motion evaluations. In Chapter 3 I present my initial empirical baseline evaluation of two thumb typing on mini–QWERTY keyboards. In this chapter I outline a method and participant compensation strategy that becomes a standard for my future studies. I present the results and discuss the implications of these results comparing performance on mini–QWERTY keyboards to performance on other text input technologies. In Chapter 4, I discuss a study designed to investigate the impact of inputing text in conditions of limited visual feedback. I contextualize blind mobile text input and further discuss my methodology, results, and the implications of the study. In Chapter 5, I present my mobility evaluation in which I investigate the impact of walking, sitting, and standing on individuals' ability to input text while on–the–go. In Chapter 6 I analyze the types of errors that individuals make when typing on mini–QWERTY keyboards and discuss my solution for automatically detecting and correcting those errors in Chapter 7. In Chapter 8 I describe the final version of my error detection algorithm and discuss how I arrived at a simplified approach for detecting and correcting errors in mini-QWERTY typing. Finally, in Chapter 9 I discuss my final study which is designed to gather human performance metrics of individuals typing using my automatic error correction software. The dissertation concludes with Chapters 10 and 11 in which I summarize the dissertation, discuss future work, and conclude. Table 1 presents a summary of the research questions, hypotheses, methodology, data collected, and publication status for each study.

Study	Research Question	Hypothesis	Method	Data Collected	Status
Baseline	How quickly and accurately can individuals input text on mini-QWERTY keyboards?	Mini-QWERTY keyboard typists learn quickly and type both more quickly and accurately than any other standard mobile text input methods recorded in the literature	To assess empirical baseline human performance measures of typing on mini-QWERTY keyboards I conducted a longitudinal laboratory-based study. Recruiting 14 participants, I assessed typing performance over twenty 20-minute typing session.	Word–per–minute (WPM) and accuracy (ACC) measures for 14 participants over 400 minutes of use.	**CHI05**
Blind	How quickly and accurately can expert mini-QEWRTY typists input text on mini-QWERTY keyboards in conditions of limited visibility?	Conditions of limited visibility will dramatically impair the typing rates and accuracies of mini-QWERTY keyboard typists	To assess the impact of typing in conditions of limited visibility, I conducted a longitudinal laboratory-based study. I trained eight participants to expertise, I then assessed their typing performance in three different visibility conditions over five 20-minute sessions.	WPM and ACC measures for eight participants typing in three different visibility conditions for 100 minutes each.	**ISWC05 (Best-paper nomination)**

Continued on next page

Table 1 continued from previous page

Study	Research Question	Hypothesis	Method	Data Collected	Status
Mobile	What is the impact of mobility on individual's ability to input text on mini-QWERTY keyboards?	Mobility will have a significant negative impact on participants' typing performance.	Participants type for 300 minutes while seated in the lab to train up to expertise. The participants are then divided into groups and balanced across three mobility conditions (sitting, standing, and walking) using Latin square balancing. They then proceed to type for 100 minutes per mobility condition.	WPM and ACC measures for 36 participants over 300 minutes of "mobile" use. Collect distance traveled and speed	**to be submitted to CHI'14**
Automatic Whiteout	Can we leverage features of the users' typing to automatically detect and correct typographical errors made on mini-QWERTY keyboards?	Leveraging features of the users' typing we can automatically detect and correct more than 20% of the typing errors on mini-QWERTY keyboards regardless of the condition in which the user is inputting text.	Analyze the types of errors that were made in the Empirical, Blind, and Mobile studies. Develop an algorithm that targets common classes of errors from each study. Theoretically validate the algorithm.	Classification of errors over a set of data that is comprised of over 300 man–hours of typing from 50+ participants.	**Mobile HCI07, CHI08, to be submitted to CHI'14**

Continued on next page

Table 1 continued from previous page

Study	Research Question	Hypothesis	Method	Data Collected	Status
Live Error Correction Evaluation	What is the impact of automatic error detection and correction on live mini-QWERTY keyboard text input?	Users with our error correction algorithm enabled will type faster with fewer errors.	Nine novice participants will be trained to expertise (300 minutes). After being trained to expertise, each participant will type 100 phrases with error correction turned on and 100 phrases with it turned off. Phrases with and without the algorithm will be randomly assigned and balanced at the block level.	WPM and ACC measures (both corrected and uncorrected) for 9 participants typing 100 phases per condition.	**to be submitted to CHI'14**

Table 1: Summary of research questions

CHAPTER II

RELATED WORK

Text entry research has a long history and predates the creation of Human Computer Interaction as an area of research and study. Historically, text input evaluation and the investigation of typographical errors have been subjects of interest to the HCI research community [16, 20]. With the explosion of interest in mobile computing over the past decade, text entry evaluation has again gained traction with the CHI community. An increased focus on text entry research has emerged due to the constraints imposed by inputting text quickly and accurately on small devices. Zhai et al. identify traits that constitute effective text input interfaces and discuss text input methods in depth [62]. Wobbrock details text entry measures at length [59] while MacKenzie and Soukorff provide a detailed discussion of challenges of analyzing errors in text input [35]. Readers are directed to these sources for more details.

This section discusses the related work in text entry for button-based mobile keyboards. The first subsection describes various mobile input systems. The following subsection describes the research exploring the challenges of inputting text while on-the-go, and this chapter concludes with a summation of research analyzing errors in mobile text input and provides a list of metrics used in this dissertation.

2.1 Mobile Keyboards

There are numerous mobile keyboard options available for entering text, and one of the most prevalent text entry devices is the mobile phone keypad. On a mobile phone keypad multiple letters are mapped on to a single key on the number pad (for example, the letters "a", "b", and "c" are printed on the number "2" key). Two common ways to enter text on a mobile phone are multi–tap and T9. To type a character using multi–tap, the user cycles through the letters assigned to a key by pressing the button multiple times. For example, if a user wishes to type the letter "s" using multi–tap on a standard mobile phone keypad, she will press the "7" key three times; to input the word "truck" she will type "8-7-7-7-8-8-2-2-2-5-5." Novice multi–tap users begin typing at approximately 8 words per minute (wpm) [5, 23] while experienced users reach speeds in the 16–20 wpm range [31, 34].

T9 is another common mobile phone entry text method. Unlike multi-tap, the user only presses each key containing the desired character once. T9 computes all of the permutations of letters that correspond to the sequence of keypresses entered by the user and employs a dictionary to disambiguate the results. It then presents a list of words to the user with the most likely word appearing first in the list. For example, to input the word "truck" using T9 a user would type "8-7-8-5-2." T9 displays the most commonly used word with the "8-7-8-5-2" number sequence: "usual". In this case, "truck" is the second most commonly used word with that number sequence. In order to get T9 to correctly output the desired word, the user must select it from the list presented. In cases where the user inputs a word that is not in the dictionary, the user is asked to input the word a second time using multi–tap. She then has the option of adding that word to the T9 dictionary for all future lookups. T9 typing rates range from 9 wpm for novices to 20 wpm for experts [23].

Recently, several new methods have been developed for entering text on mobile phone keypads including LetterWise [34], TiltText [57], and ChordTap [58]. These methods offer novice performance similar to multi–tap (7.3 wpm, 7.4 wpm and 8.5 wpm, respectively). In addition, each of these methods offer rapid expert typing rates. LetterWise users achieved a rate of 21.0 wpm after approximately 550 minutes of practice. TiltText users reached 13.6 wpm, and ChordTap users achieved 16.1 wpm with about 160 minutes of typing practice.

In addition to the standard mobile phone keypad, there are a host of other physical keyboards designed to facilitate mobile text entry. Some examples of these are the Half–QWERTY chording keyboard [39], the Twiddler one–handed chording keyboard [30], and the mini–QWERTY keyboard.

The Half–QWERTY chording keyboard is one half of a desktop–QWERTY keyboard (full sized keys, traditional QWERTY key layout) requiring only one hand to input text. The wearable version of this keyboard is designed to be worn on the wrist of the non–typing hand. Instead of pressing keys in sequence to produce a character, a chording keyboard enables multiple keys to be pressed simultaneously to generate a character (or in some cases, a series of characters). For the Half–QWERTY chording keyboard, in order to generate keys on the "missing" half of the keyboard, the user holds down the space bar and types normally using the remaining four fingers. While the space bar is being held, the keys are mapped to the letters on the missing half corresponding to their mirrored values (e.g. G→H, F→J, D→K, S→L, A→;, etc.). In experimental conditions, participants were able to become familiar with the Half–QWERTY keyboard quickly. After ten hours of practice, ten participants typed between 23.8 to 42.8 wpm, which was 41% to 73% of their two-handed speeds [39].

The Twiddler is a mobile one–handed chording keyboard with a keypad similar to a mobile phone. It has twelve keys arranged in a grid with three columns and four rows on the front. Unlike a mobile phone, the Twiddler is held with the keypad facing away from the user, and each row of keys is operated by one of the user's four fingers. After 25 hours of practice, Twiddler users are able to type 47.3 wpm on average [30].

2.2 Mini–QWERTY Keyboards

The mini–QWERTY keyboard is a mobile two–handed keyboard with a keypad similar in size to that of a mobile phone keypad. It contains one key for each letter and is configured similarly to a desktop QWERTY keyboard, complete with space, delete, enter, and other non–letter keys. It is a handheld keyboard and is typically operated

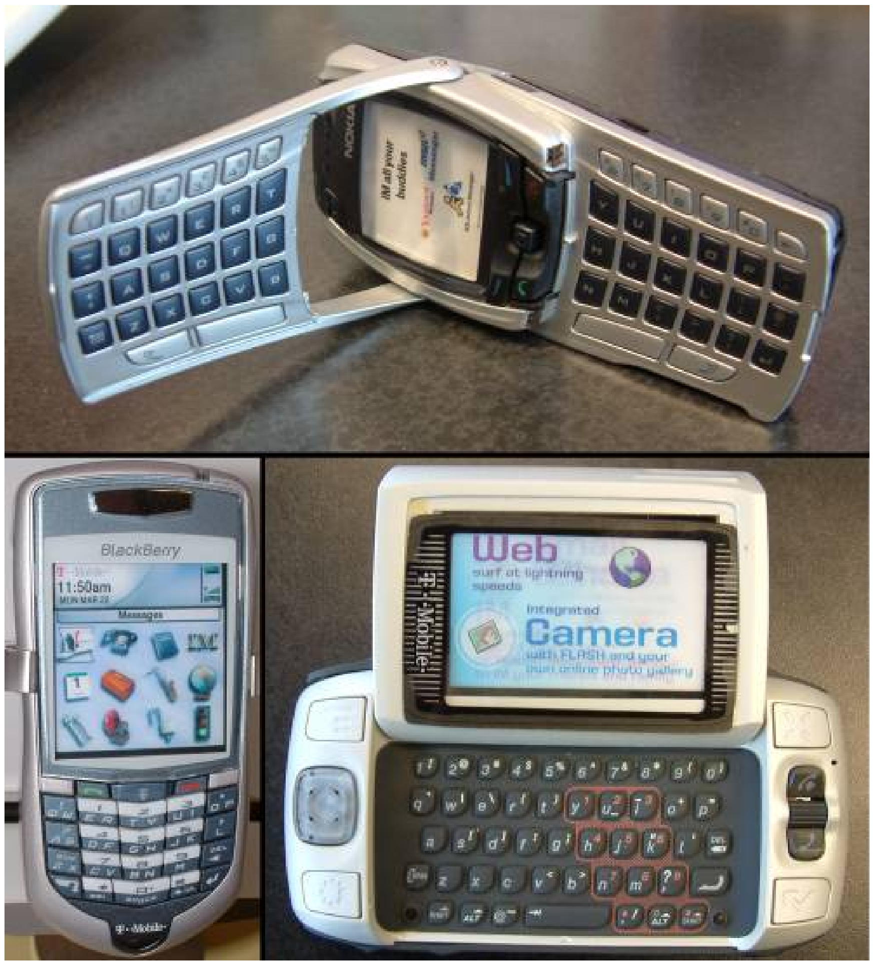


Figure 1: Commercial mobile phones with mini–QWERTY keyboards: Nokia 6820 (top), RIM Blackberry (bottom left) and Danger/T–Mobile Sidekick (bottom right).

using only two thumbs.

Several examples of commercial mini–QWERTY devices are shown in Figure 1. Nokia has taken a somewhat non–traditional approach with its 6800 series of mobile phones. Its front face can flip open to reveal a split mini–QWERTY layout, with the screen set in the middle of the keyboard. The Research In Motion (RIM) Blackberry mobile information device has included a mini–QWERTY keyboard since 1999. The Danger HipTop (also known as the T–Mobile Sidekick) is a similar device which includes a mini–QWERTY keyboard under a flip–up screen. For quite some time it was popular with younger demographics and with the Deaf community because of its combination of mobile phone, mobile e-mail, web browsing, and instant messaging capabilities [22].

Despite the presence of mini–QWERTY keyboards in the mobile computing marketplace, there is relatively little published research on user typing rates with these devices. Researchers at Canesta, Inc. produced a study that included mini–QWERTY typing speeds [48]. In evaluating their virtual projection keyboard, they tested it against a desktop QWERTY keyboard, Graffiti pen input, and a mini–QWERTY keyboard. They recruited 11 subjects who used each method in random order, typing a single phrase repeatedly for 2 minutes. Subjects achieved an average of 27.6 wpm on the thumb keyboards, 64.8 wpm on the conventional keyboard, 46.6 wpm on the Canesta keyboard, and 14.0 wpm with Graffiti. The authors state their participants included both novice and expert Canesta keyboard users but do not mention participants' experience with any of the other input devices.

In addition to the Canesta study, Mackenzie and Soukoreff have created a theoretical model of two–thumb text entry on miniature keyboards [36]. Using English language letter frequency distributions and Fitts' Law calculations, they predicted an expert typing rate of 60.74 wpm on a mini–QWERTY layout. A sensitivity analysis of the model to various parameters (e.g., Fitts' Law coefficients) yielded no more than a +/- 10% variation from the original figure. In Clarkson et al. we discuss this model in the context of results presented in this dissertation [10].

2.3 Mobile Text Entry On-the-Go

This section discusses previous work focused on the evaluation of mobile interaction, interfaces, and devices while users are in-motion with a special emphasis placed on investigations of on-the-go text entry.

2.3.1 Controlled Use–In–Motion Studies

Recently there has been increased interest in evaluating the use of mobile devices (such as mobile phones or on-body interfaces) while the user is in motion. Designing for mobility has been highlighted as an important area in need of additional research for years [2, 4, 14, 24, 42, 45]. Mobile devices create opportunities for users to complete certain tasks (e.g., taking notes or scheduling appointments) on the move, but they also introduce new challenges by creating competition for a user's attentional resources. This competition forces the user to split their attentional resources between interacting with the device and navigating the environment [44]. Interacting with mobile devices under stationary settings is similar to the use of a desktop computer in terms of attention allocation. However, when the user is mobile, attentional demands increase.

My research is focused on understanding the impact that motion has on interactions with mobile computing devices, specifically inputting text using mini–QWERTY keyboards. Such interactions rely on two fundamental activities: retrieving the information shown on the screen (output) and entering information (e.g., commands or text) into the device (input). To date, researchers have studied the impact of walking on both input– and output–oriented activities. For example, Barnard et al. [1] report that walking affects performance for information retrieval tasks including both reading comprehension tasks and word search tasks. Similarly, Price et al. [46] confirmed that walking can significantly affect the accuracy of speech–based input. Vadas et al. furthered Barnards' work by decoupling input from output and comparing visual and audio output while in motion [56].

Hall et al. compared target selection accuracy on finger–operated touch screens under seated and standing conditions [21]. They found that in order to achieve greater than 99% accuracy, the target area for the standing condition (30 mm^2) had to be larger than that of seated condition (26 mm^2). Although standing is not a mobile condition, it shares one key characteristic with mobile situations: the lack of a stable surface on which to place the mobile device. Schildbach et al. further explored finger–operated touch screen input in a study that examined target acquisition and reading tasks with users both standing and walking [49]. They found negative effects for both reading and selecting targets when walking although the negative effect associated with target selection were mitigated when they increased the size of the targets. Interestingly, the reading task did not see the same compensatory effect with larger text size since increasing the size of the text increased the need for scrolling. This result was not surprising given that MacKay et al. had earlier compared using scrollbars to two other navigation techniques (tap-and-drag and touch-and-go) in three different mobility conditions (sitting, standing, and walking) and found scrollbars to be the poorest performing navigation technique for mobile users [32].

Brewster [2] explored use–in–motion in an early multimodal study. He investigated the use of a stylus-based calculator application under two conditions: seated in a lab and walking outside. The results showed that walking significantly reduced the amount of data entered and increased the perceived workload, indicating some difficulties associated with stylus–based tapping while walking. The walking route used in this study was a natural setting, but as the researcher himself pointed out:

> "[It] was still quite controlled: users walked along a reasonably quite straight path [2]."

A similar, simple, and safe walking condition was also used in a follow–up study [3]. The decline in performance from seated to walking may have been underestimated by these results due to the simplicity of the walking task. In addition, this study did not collect any error data, which makes it difficult to interpret task completion time and workload results.

In contrast, Chamberlain and Kalawsky [6] did not find a significant difference in error rates between stationary and walking conditions when participants completed stylus–based target selections. They did, however, find an increase in selection times when participants were walking. In their study, a wearable computer was attached to a vest which provided additional support. This support made the computer more stable than one could expect to occur in more common use-in-motion situations in which an individual simply holds the device in their hand. This added stability may help explain why their results differed from those reported elsewhere. Lin et al. also investigated mobile pointing using a stylus while walking [28, 27]. In their first study Lin et al. had participants complete a Fitts' law pointing task using a stylus while walking. They found impaired performance while walking when targeting small targets [28]. A follow-up study by Lin et al. found that single target selection completion time did not increase while walking, but that overall task completion times did increase as did subjective ratings of cognitive load [27].

Kane et al. examined use–in–motion in an outdoor evaluation and coined the term "walking user interfaces" [25]. They evaluated an adaptive interface that scaled button sizes based on user motion. They found significant interactions between size and movement and demonstrated that dynamic user interfaces performed as well as the equivalent static interfaces without any additional penalty due to adaptation.

2.3.2 Typing on-the-go

Although there has been much work examining both input and output while users are walking, text entry while in motion is an underexplored research area. Mizobuchi and Yatani both investigated touchscreen text input using a stylus [41, 61]. Mizobuchi et al. studied the relationship between walking speed and text entry difficulty [41]. They examined four different key sizes ranging from 2.0 X 2.5 to 5.0 X 6.3 mm. Participants entered text using a soft keyboard while either standing or walking. The results showed that text input speed did not differ between the standing and walking conditions. This lack of difference between mobility conditions might be due to the simple–and–safe walking setting used, which was comparable to those in Brewster [2] and Brewster et al. [3]. However, error rates were significantly different between the two conditions, but only because of the high error rate for the smallest target size in the walking situation.

In contrast, Yatani et al. explored the design of a two-handed virtual keyboard for a PDA which utilized the thumb of the hand holding the device to create a chorded input system [61]. They compared their chorded keyboard to other stylus-based text entry methods while participants were seated and walking around a path in the lab. Of note, Yatani et al. reported a difference in performance (WPM, accuracy, and cognitive load) between stationary and mobile conditions when participants typed on a virtual mini-QWERTY keyboard with a stylus. Most recently Goel et al. designed and evaluated an adaptive system for two-thumb text entry on touch screens that leveraged the phone's built-in accelerometer to compensate for the vibrations that result from walking. Their system was shown to both reduce errors and increase typing rates on touch screen mobile phones [17].

2.4 Text Entry Evaluations

In the past decade, unconstrained text entry studies (experiments that allow participants to commit typographical errors and have the choice to correct or not correct them) have become the accepted method for studying text input in the HCI community. The adoption of unconstrained text entry studies by the community is, in large part, due to the success of research on text input error classification and analysis. Having put considerable effort into classifying and accounting for errors, the text entry community has recently arrived at a consensus on how to handle errors in text entry studies.

Typically when testing a text entry method, short phrases are displayed one at a time to participants who transcribe each phrase into an experimental software. The output from these studies includes the presented text string (the phrase displayed to the participant), the transcribed text string (the final text produced by the participant), and the input stream (a record of all input events generated by the participant).

“Constrained” text entry studies force participants to enter text in such a manner that the transcribed text perfectly matches the presented text. This practice results in participants typing at 100% accuracy but, since participants are forced to correct all of their mistakes, it yields much slower, and less externally valid, typing rates. Until the recent work on understanding typographical errors, conducting constrained experiments was a safe way of getting results that were uncomplicated by errors.

Over the past few years, improvements in text entry error rate measurement have afforded researchers the ability to conduct “unconstrained” text entry studies (studies in which subjects are allowed to correct or not correct errors while typing). Until now, efforts have focused on how to identify and classify errors input by the user whether corrected or left uncorrected. The disambiguation and measuring of errors in unconstrained text entry experiments has received considerable attention from the CHI community [18, 35, 52, 53, 54, 60] and in recent years, unconstrained studies have become the preferred study design methodology.

2.5 Accuracy Measurements in Text Entry Studies

In what has become the standard procedure for a text entry experiment, participants are presented with different methods of inputting text in either a within-subject or between-subject design. Most studies compare performance across typing methods, reporting speed and accuracy results from their participants. To ensure that collected data are true baseline human performance measures, text input studies typically involve transcription typing since having participants generate their own messages would not yield comparable results across participants. Short phrases (usually lacking capitalization, punctuation, numbers, or symbols) are displayed to the participant one at a time by the software used to conduct the study. These phrases are often sampled from the MacKenzie and Soukoreff phrase set [37] and are presented to the participant in a random order. Participants enter each phrase using the text entry method under investigation, generating data in the form of a set of presented strings, input streams, and transcribed strings.

A benefit of allowing errors to be committed and analyzing them in this manner is that researchers can study not only errors but also corrections. Methodologically, this approach is preferred because it mimics text entry as it occurs in typical usage. Participants are instructed to "enter the presented text as quickly and accurately as possible," and they are allowed to both incur and correct mistakes. However, allowing participants to choose whether or not to correct their errors introduces ambiguity resulting in two classes of errors, those that the subject commits and then corrects (corrected errors), and those that the subject commits and either does not notice or chooses not to correct (uncorrected errors). These uncorrected errors therefore remain in the transcribed text string. Until recently, errors were identified and calculated by comparing the transcribed string to the presented string using the Minimum String Distance (MSD) Error Rate [52]. However, the MSD error rate only measures the uncorrected errors. The corrected errors are not present in the transcribed text and as such are not reported when using MSD. Soukoreff et al. proposed using the average keystrokes per character (KSPC) as a dependent variable to capture the overhead due to errors committed and correct while a user is typing [53]. While useful, this metric conflates the difficulty of error correction, user "carefulness" (number of errors the user corrects), and other specifics of the input method.

In an effort to address this weakness, Soukoreff et al. went on to describe an updated approach to measuring errors that was consistent with their previous work, yet had better dependent measures for both corrected and uncorrected errors [53]. They followed that effort with yet another paper in which they presented a metric that considered the input stream as well as the transcribed text [54]. In that analysis Soukoreff et al. were able to delineate participants' keystrokes into four classes:

- Correct (C) keystrokes – alphanumeric keystrokes that are not errors,

- Incorrect and Not Fixed (INF) keystrokes – errors that go unnoticed and appear in the transcribed text,
- Incorrect but Fixed (IF) keystrokes – erroneous keystrokes in the input stream that are later corrected, and,
- Fixes (F) – the keystrokes that perform the corrections (i.e., delete, backspace, and cursor movement).

Having separated keystrokes into the above four groups, Soukoreff et al. presented the following measures [53]:

$$TotalErrorRate = \frac{INF + IF}{C + INF + IF} \cdot 100\% \tag{1}$$

$$CorrectedErrorRate = \frac{IF}{C + INF + IF} \cdot 100\% \tag{2}$$

$$NotCorrectedErrorRate = \frac{INF}{C + INF + IF} \cdot 100\% \tag{3}$$

To address the circumstance where a user has to delete correct keystrokes while attempting to address an error that occurred earlier in the input stream, Soukoreff subdivided the "Incorrect but Fixed" category into two groups: IF keystrokes that were correct (IFc) and IF keystrokes that were errors (IFe). Doing so created two new error types [54]:

$$CorrectedButRightErrorRate = \frac{IFc}{C + INF + IF} \cdot 100\% \tag{4}$$

$$CorrectedAndWrongErrorRate = \frac{IFe}{C + INF + IF} \cdot 100\% \tag{5}$$

Wobbrock et al. expanded this work through the presentation of a taxonomy of input stream error types [60]. They clarified the error classes introduced by Gentner et al. [16] and MacKenzie et al. [35] by naming them *Uncorrected No–Errors, Uncorrected Substitutions, Uncorrected Insertions*, and *Uncorrected Omissions.* They then introduce the *corrected* character–level error types:

- *Corrected No–Errors* Characters that are input correctly by the user but erased
- *Corrected Substitutions* Erroneous characters input by the user in an attempt to correct an existing error.
- *Nonrecognition Substitutions* "Nonrecognition substitutions occur when an attempt to produce a character yields no result [60]." Though primarily occurring when a user employs a stroke–based text entry method such as Graffiti, these errors may occur when evaluating fixed–key text entry techniques if the keypad has extraneous buttons that do not generate a valid output when pressed.
- *Corrected Insertions* Erroneous characters inserted into the input stream that are then noticed and corrected by the user. Corrected insertions exist in the input stream but not in the final transcribed text.
- *Nonrecognition Insertions* Unrecognized characters inserted at the end of a word when each letter of the input stream has already been paired with a letter of the presented text.
- *Corrected Omissions* Corrected omissions occur in the input stream when the user initially skips a character but then later replaces the omitted character correcting the omission.

2.6 Measuring Speed in Text Entry Studies

In contrast to challenges of calculating accuracy and error rates in unconstrained studies, calculating speed in text entry evaluations is relatively simple. The common measure for speed in text entry evaluations is words-per-minute (WPM). In this dissertation, words-per-minute rates are calculated using the traditional approach outlined by Mackenzie [33] (see Equation 6) where T is the length of the transcribed text and S is the time it takes to enter the entire phrase in seconds. The constants 60 and $\frac{1}{5}$ are used since there are 60 seconds in a minute and the average length of a word (including spaces) is five characters.

$$WPM = \frac{T-1}{S} \cdot 60 \cdot \frac{1}{5} \tag{6}$$

When reporting the results of the longitudinal evaluations in Chapters 3, 4, and 5, I discuss both WPM (Equation 6) and Accuracy measures. Accuracy is calculated using Equation 7.

$$ACC = 1 - TotalErrorRate = (1 - \frac{INF + IF}{C + INF + IF}) \cdot 100\% \tag{7}$$

In Chapter 6, there is much discussion of the Wobbrockian character level error types. Chapters 7, 8, and 9 refer to all of the above metrics.

CHAPTER III

AN EMPIRICAL STUDY OF TYPING PERFORMANCE ON MINI–QWERTY KEYBOARDS

In the Fall of 2004, we conducted our initial mini–QWERTY keyboard text input study. This study was designed to ascertain novice mini–QWERTY keyboard typing rates and accuracies. It also allowed us to compare desktop QWERTY performance to typing on a mini–QWERTY keyboard and examine learning rates to determine the length of time it takes to achieve expertise. This chapter details our study methods (Section 3.1), presents participant demographics (Section 3.1.1) and describes our results. In Sections 3.3 and 3.3.1 we discuss both the quantitative and qualitative results of our evaluation and identify some of the major findings.

3.1 Method

A number of aspects of our study design are directly influenced by previous research. In particular, the compensation arrangement and session structure follow those of previous keyboard studies by Lyons et al. [30] and MacKenzie et al. [38, 34].

3.1.1 Participants

We recruited 21 participants who had not used a mini–QWERTY keyboard more than once. All were experienced full–QWERTY keyboard users. Each participant was randomly assigned one of two different mini–QWERTY keyboards to use throughout

This chapter is an excerpt from Edward Clarkson, James Clawson, Kent Lyons, and Thad Starner, "An Empirical Study of Typing Rates on mini–QWERTY Keyboards," in *the extended abstracts of CHI, 2005* [9] the study (Dell and Targus branded, discussed further below). Participants were compensated proportional to their typing rate and accuracy over the entire session: $0.125 $\times$ wpm $\times$ accuracy, with a $4 minimum for each twenty minute session.

Participants were asked to complete 20 twenty-minute sessions over the course of 11 days. Four participants did not complete all 20 sessions, leaving 10 participants in the Targus group and 7 in the Dell. To ensure a balanced study design for our analysis, we excluded the data from the last three participants in the Targus group to have signed up for the study in order to have the same number of participants (7) in both groups. These 14 participants ranged in age from 18 to 34. Six participants were female, eight male, and only one was left–handed.

3.1.2 Equipment and Software

Figure 2 shows the two mini-QWERTY keyboards used in the experiment, one by Dell (for the Dell Axim) and one by Targus (for the Palm m505). We modified each keyboard to connect to a standard desktop computer serial port. The Dell and Targus keyboards transmit at 4800 and 9600 baud, respectively.

The letter keys on both keyboards are oval–shaped as shown in Figure 2. The Targus keys measure 6.73 mm along their major axes and 4.83 mm along their minor axes. The Dell keys are 5.99 mm along the major axes x 4.06 mm along the minor axes. Furthermore, the Dell keyboard has a single space key in the middle of the bottom key row, while the Targus has two triangular space keys set below the rest of the keys.

The study occurred in our usability lab with each of the two keyboards connected to a separate Pentium III workstation. We employed the Twidor typing software package (used in our previous series of studies on the Twiddler chording keyboard [30]) and adapted it to accept data from our modified keyboards. The Twidor software was configured to use the MacKenzie and Soukoreff phrase set [37], a set of 500 phrases

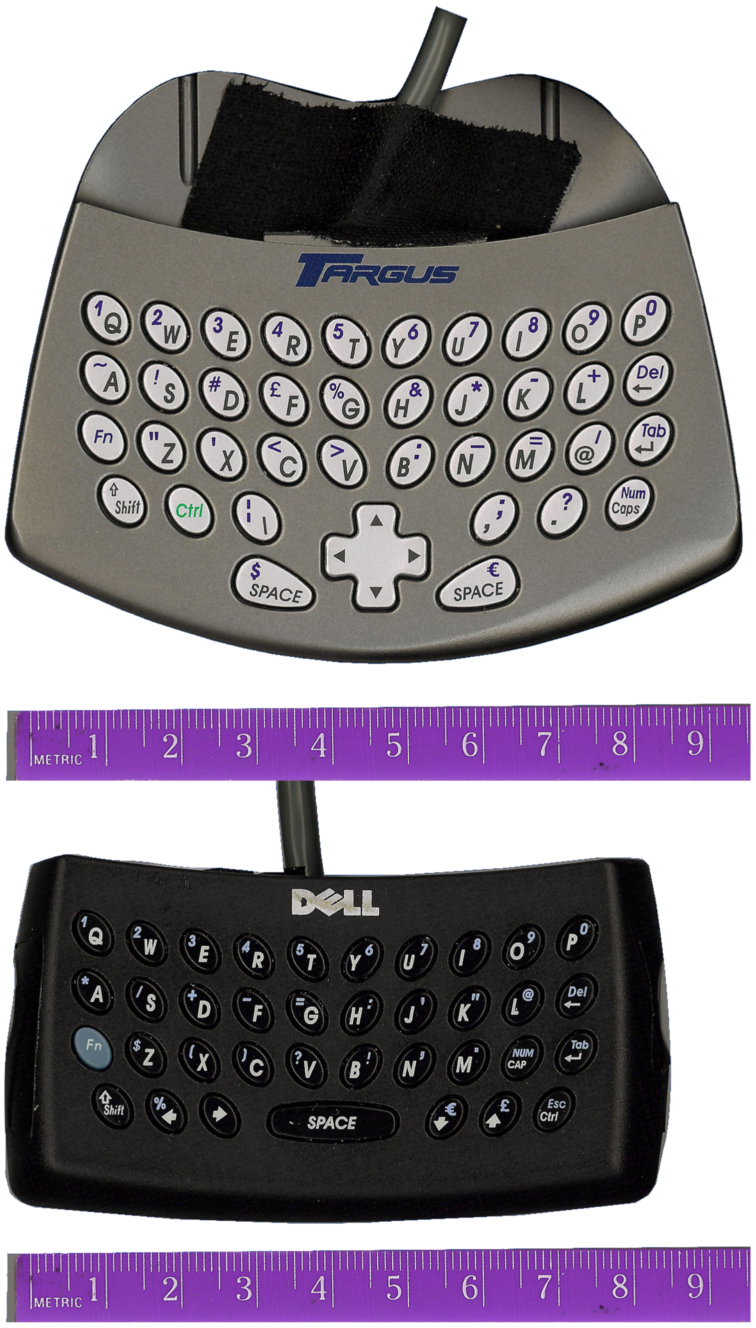
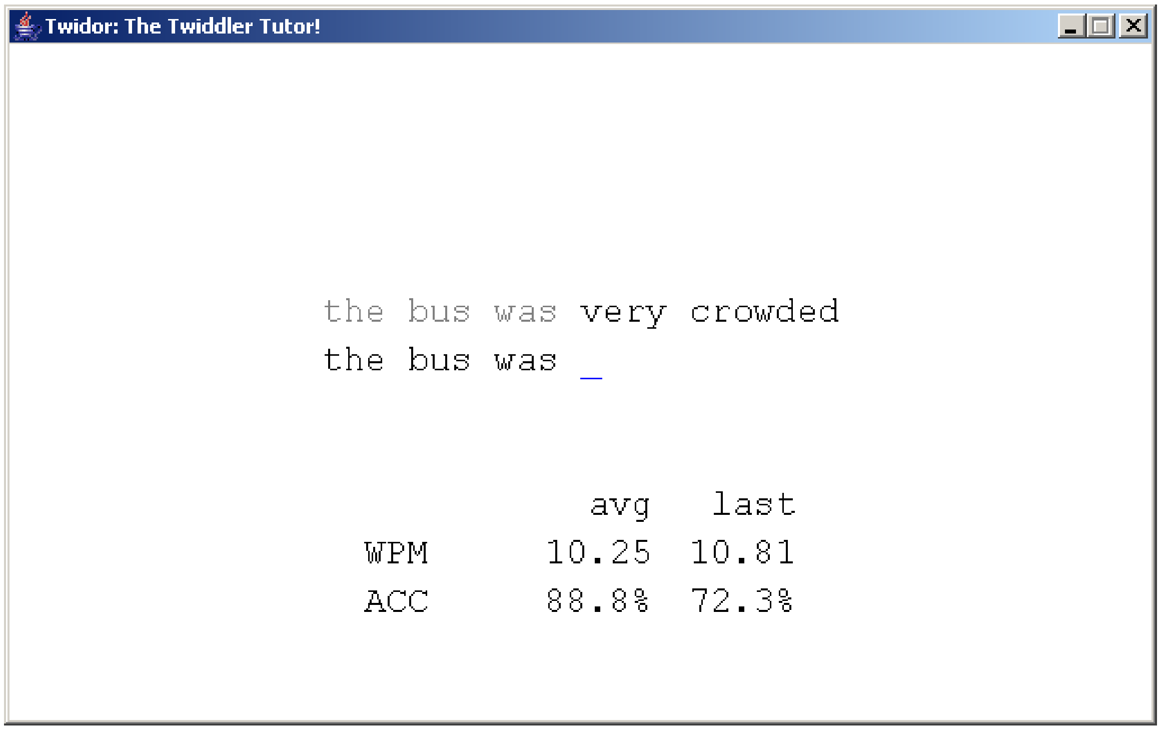


Figure 2: The mini–QWERTY keyboards used in both studies: Targus (top) and Dell (bottom) keyboards. Keyboards shown in the figure are actual size.



```
the bus was very crowded
the bus was _

            avg    last
WPM       10.25   10.81
ACC       88.8%   72.3%
```

Figure 3: The Twidor experimental software used in both the original and the blind studies.

representative of the English language. The phrases range from 16 to 43 characters with an average length of 28 characters. The canonical set was altered from British English spellings to use American English spellings as this study took place in the United States. Additionally, we altered the set so that there was no punctuation or capitalization in any of the phrases in the set. Twidor displays a phrase to the user. The user is asked to input the phrase as quickly and accurately as possible and Twidor displays the text produced by the participant. Twidor records the user's words per minute (WPM), and accuracy (ACC) and displays both the WPM and the ACC of the input phrase to the user as shown in Figure 3. Twidor also displays a user's average WPM and ACC calculated over the course of the session as well.

3.2 Design and Procedure

We structured the study as a 2 x 20 mixed design, with the keyboard as the between–subjects factor and the 20 sessions as the within–subjects factor. Sessions were completed in pairs with a 5–minute break after the first 20–minute session. Each session pair was separated by at least two hours and by no more than two days. Each participant was randomly assigned a keyboard in the beginning of the study that they used throughout their participation in the evaluation.

Each session was preceded by a warm–up phase, which consisted of the phrases "abcd efgh ijkl mnop" and "qrst uvwx yz" repeated twice. The warm–up phrase was not counted in the session statistics. The remainder of the session consisted of a number of trial blocks each containing ten randomly selected phrases. The participants completed as many blocks as he or she could in a twenty–minute session. The participants were instructed to type using only their two thumbs and to type as quickly and accurately as possible. The test software provided statistical feedback in the form of typing rate and accuracy data for the most recent sentence typed and the current session average.

In addition to the mini–QWERTY rates, we also collected desktop QWERTY typing speeds averaged over 20 phrases at each participant's first and twentieth sessions. Participants also completed a demographics survey before the first session and a debriefing survey after the end of the last session.

3.3 Results

The 14 participants typed 33,945 phrases across all sessions, encompassing over 950,000 individual characters[1].

[1]Due to an error in the Targus mini–QWERTY keyboard firmware, we excluded the first five characters of each phrase for both devices. The Targus keyboard sent a wake–up call on the first key–down event after a period of inactivity. While waiting for the wake–up call to generate a response, the keyboard buffers approximately two keypresses. As our participants started reaching expert rates, they began to exceed the two character buffer. For that reason, results presented in this paper

Table 2 shows the typing rates for the two keyboard groups during the first and last session. Averaged over both keyboards, participants had a mean first session typing rate of 31.80 wpm (SD = 5.59) and a session twenty mean of 59.03 wpm (SD = 8.55). The average accuracy for session one was 95.12% (SD = 2.30%), declining gradually to 92.99% (SD = 3.71%) by session twenty. Table 3 shows the accuracies for the two keyboard groups during the first and last session.

Keyboard Group	**Session 1**	**Session 20**
Targus	34.27 wpm ($SD = 6.32$)	58.74 wpm ($SD = 7.46$)
Dell	29.33 wpm ($SD = 4.86$)	59.32 wpm ($SD = 9.65$)
Mean	31.80 wpm ($SD = 5.59$)	59.03 wpm ($SD = 8.55$)

Table 2: Mean mini–QWERTY typing rates and standard deviations for Session 1 and Session 20.

Figure 4 shows the typing rates per session. Overall, the Targus group typed faster than the Dell group. A two–way ANOVA (with session and keyboard type as factors) shows no interaction ($p > 0.9999$) and significant effects for both the session and keyboard ($p < 0.01$). These results indicate that the Targus group was faster than the Dell and, not surprisingly, typing speeds improved with practice.

The mean desktop QWERTY typing speed for the Targus group and the Dell group were measured before the first session and after the final session and can be found in Table 4. We measured the correlation between mini–QWERTY and desktop typing rates for both novices (Session 1) and experts (Session 20). We found correlations of $r^2 = 0.31$ for Session 1 and 0.57 for Session 20. We hypothesize that the

differ from previously reported results as the issue was discovered after publication of the original study [9]

Keyboard Group	**Session 1**	**Session 20**
Targus	4.97% ($SD = 2.07\%$)	5.32% ($SD = 2.09\%$)
Dell	4.78% ($SD = 2.53\%$)	8.71% ($SD = 5.32\%$)
Mean	4.78% ($SD = 1.47\%$)	6.17% ($SD = 3.39\%$)

Table 3: Mean mini–QWERTY error rates and standard deviations for Session 1 and Session 20.

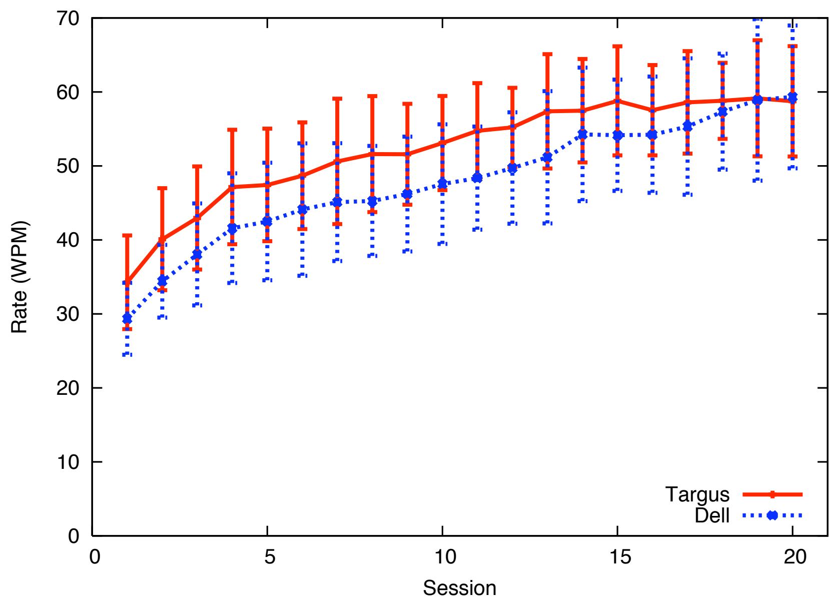


Figure 4: Per–group session average wpm with standard deviations. Dell curves are on the bottom (blue); Targus curves on top (red).

Keyboard Group	Session 1	Session 20
Dell	77.38 wpm ($SD = 15.03$)	86.99 wpm ($SD = 14.20$)
Targus	84.99 wpm ($SD = 19.84$)	98.31 wpm ($SD = 19.25$)
Mean	81.19 wpm ($SD = 5.59$)	92.65 wpm ($SD = 8.55$)

Table 4: Mean desktop typing rates and standard deviations for before session 1 and after session 20.

increase in full–QWERTY speeds is likely due to an increased familiarity with the test environment.

3.3.1 Survey Results

At the end of the final session, each participant completed a debriefing survey, answering a number of free–form and 7–point Likert scale questions regarding how they used the keyboards and their comfort level. In response to the question: “Overall, how comfortable did you think the device was to type on?” the 14 participants found the mini–QWERTY keyboards to be marginally comfortable (M = 4.00, SD = 1.41; 1 represented extremely uncomfortable, 7 extremely comfortable). When asked “How comfortable did you find it compared to a normal, full–sized keyboard?" users responded that it was less comfortable than a full–size keyboard (M = 2.29, SD = 0.99). A Student's t-test indicates no statistical difference in comfort ratings between the two mini–QWERTY keyboard groups (p = 0.36). Participants also reported they were less likely to look at the screen after the last session than at the start; however, this findingwas a marginally significant result (p = 0.056). This data conforms to some anecdotal responses (e.g., "Not looking at the screen vastly improved my speeds!").

3.4 Discussion

Overall, our results indicate that experienced desktop QWERTY keyboard users are able to quickly familiarize themselves with mini–QWERTY keyboards. The rates attained by our participates after only twenty minutes are faster than expert typing rates recorded on most other mobile keyboards (Section 2.1).

The mini–QWERTY keyboard affords rapid mobile text entry with expert users sustaining a typing rate of almost 60 wpm over the course of a twenty minute typing session. The mini–QWERTY keyboard is not just an effective tool for experts, novices also find it easy to use. Assuming the user already knows how to type on a desktop QWERTY keyboard, novice users average more than 30 wpm after only twenty minutes of practice. No other mobile text entry device affords comparable rates as quickly as the mini–QWERTY keyboard.

Though our participants demonstrated that they could type and learn quickly, mini–QWERTY keyboards may not be suitable for all mobile text entry situations. In particular, the use of a mini-QWERTY keyboard in contexts where the user is not able to focus her attention entirely on entering text, is unexplored. Understanding the effects of typing in conditions with limited visual feedback is particularly important as mobile devices are increasingly being used while individuals are on-the-go or multitasking.

CHAPTER IV

THE IMPACTS OF LIMITED VISUAL FEEDBACK ON MOBILE TEXT ENTRY FOR THE TWIDDLER AND MINI-QWERTY KEYBOARDS

Overall, our first experiment showed that our participants quickly learned how to enter text on mini–QWERTY keyboards. The typing rates compare favorably with other mobile devices for both novices and experts. However, in addition to typing rapidly and accurately, reliably entering text in a mobile environment poses several additional challenges. For example, a user cannot always look at her device in a mobile setting and may be forced into situations where she needs to enter text without being able to see either her display or the keyboard she is using. The user may be concerned with monitoring a piece of industrial equipment, looking at the ground while walking, or engaging in a face–to–face conversation and cannot focus on the computer. As such, it is important to be able to enter text with limited visual feedback.

In the following sections we examine blind typing on mini–QWERTY keyboards and introduce a taxonomy for blind mobile text input. In the Spring of 2005 we conducted a study in which eight expert mini–QWERTY typists participated in five typing sessions. Each session consisted of three twenty–minute typing conditions. In the first condition, the control or "*normal*" condition, the participant has full visual access to both the keyboard and the display. In the second condition, "*hands blind*,"

This chapter is an excerpt from James Clawson, Kent Lyons, Thad Starner, and Edward Clarkson "The Impacts of Limited Visual Feedback on Mobile Text Entry for the Twiddler and MiniQWERTY Keyboards," in *the proceedings of ISWC, 2005*[12] we obstruct the view of the keyboard. The final "*fully blind*" condition also reduces visual feedback from the display.

4.1 Blind Typing in Real Environments

Typing with limited visual feedback of the keyboard or display can happen in many different mobile situations. One study by Silfverberg evaluated the abilities of his participants to quickly and accurately enter text without looking at the keyboard or display [51]. One motivation for this study was observations of Finnish students secretly sending text messages to each other in the classroom. The students would place their mobile phones under the desk and input text without the device being visible. However, sending text messages discretely is not the only condition in which a user experiences conditions of limited visual feedback. Wearable computer users attempting to enter text while in face–to–face conversation also experience a similar phenomenon [29]. Blind text entry also frequently occurs in environments in which the user's attention is fragmented and the user can not attend fully to the task of inputting text. For example, inputting text while walking forces the user split her attention between her mobile device and navigating the environment.

4.2 A Taxonomy for Blind Mobile Text Input

To varying degrees, previous typing studies have explored different aspects of blind typing. Here, we present a taxonomy of blind typing conditions and describe how our studies fit within this taxonomy. We denote the output of the computer display showing the text being entered as *on–screen feedback* which is subdivided into three categories: present, limited, and absent. We have named the feedback obtained by looking at the input device *keyboard visibility*, and it shares the same three categories. Table 5 shows our taxonomy populated with previous work as well as the conditions for our blind typing mini–QWERTY study presented in the following sections.

Blind Typing	On–screen feedback Present	Limited	Absent
Keyboard visibility present	• Mini–QWERTY Normal • Silfverberg Direct–Visual Feedback	• Desktop hunt–and–peck	
Keyboard visibility limited	• Twiddler Normal	• Twiddler Dots	• Twiddler Blind
Keyboard visibility absent	• Mini–QWERTY Hands Blind • Silfverberg Indirect Visual Feedback • Desktop touch–typing	• Mini–QWERTY Fully Blind	• Silfverberg No Visual Feedback

Table 5: Taxonomy of different blind typing conditions.

Silfverberg examined the effect of visual and tactile feedback on a user's ability to successfully navigate a mobile phone keypad [51]. The 2 x 3 study explored the physical affordances of two different phone keypad layouts in three conditions of varying visibility (direct visual feedback, indirect visual feedback and no visual feedback). In the direct visual feedback condition, the participant could see the phone and receive feedback from the display (on–screen feedback and keyboard visibility is present). In Silfverberg's indirect visual feedback condition, the subject placed her hand holding the phone under the desk occluding visibility of the phone keypad. She received feedback from the display after pressing a key to indicate which key had been pressed (on–screen feedback present, keyboard visibility absent). The no visual feedback condition mirrored the indirect visual feedback condition except that the feedback from the display was removed (on–screen feedback and keyboard visibility absent). Silfverberg's study found that limited visual feedback combined with low tactile feedback increases a user's average error rate. On the other hand, good tactile feedback results in a smaller decrease in accuracy.

In previous work with the Twiddler, Lyons et al. examined blind typing [29]. As the natural hand position for the Twiddler is with the keys facing away from the user, they only evaluated the effect of changing the on–screen feedback across conditions. Their blind study had 3 conditions (normal feedback, dots feedback, and blind). The normal feedback condition displayed the text as it was typed (on–screen feedback present, keyboard visibility limited). For the dots condition, we displayed periods for each character typed instead of the transcribed text. Thus, participants see their position in the supplied phrase but not specifically what they type (on–screen feedback limited, keyboard visibility limited). This condition was designed to simulate monitoring text typed without being able to actually read the letters. This condition was inspired by the wearable computing situation where a head–up display is being monitored in the user's peripheral vision while typing notes with a mobile keyboard and maintaining eye contact with a conversation partner. Finally, the blind condition did not show any on–screen indication of what was typed (on–screen feedback absent, keyboard visibility limited). For both the dots and blind conditions, participants were shown their transcribed text and error statistics when they pressed enter at the end of a phrase. The results from the Twiddler study indicated that participants performed just as well or slightly better in the reduced visual feedback conditions [30].

In our second mini–QWERTY study we investigate this type of blind typing. Our participants type in conditions of limited visibility both of the keyboard and the display. In the first condition, the *normal* condition, the subject has full visual access to both the keyboard and the display (keyboard visibility present, on-screen feedback present) as shown in Figures 5 and 7. The second condition, *hands blind*, obstructs the view of the keyboard but presents the visual feedback normally (keyboard visibility absent, on-screen feedback present) as shown in Figures 6 and 7. The final condition, *fully blind*, not only obstructs the view of the keyboard but also reduces visual feedback from the display (keyboard visibility absent, on-screen feedback limited) as shown in Figures 6 and 8.

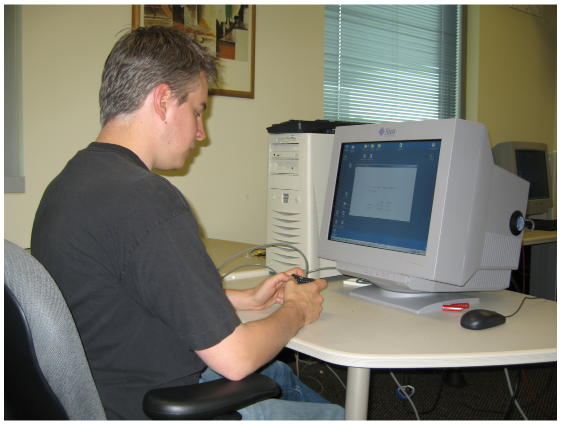


Figure 5: Experimental configuration for normal condition where the user can see the keyboard while typing (keyboard visibility present).

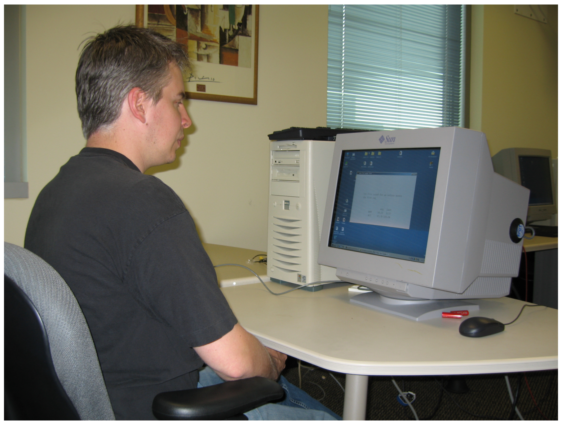
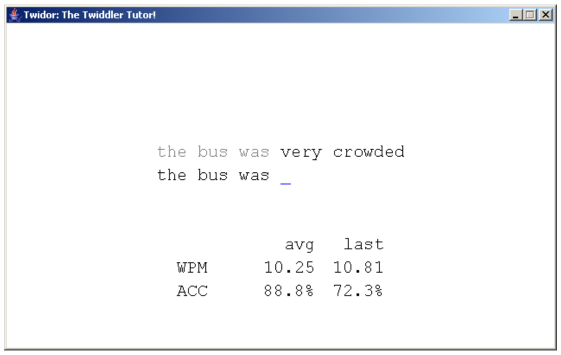


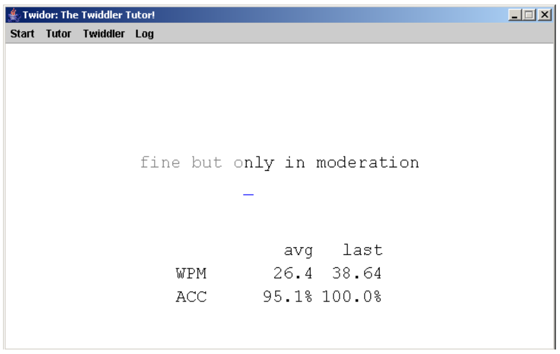
Figure 6: Experimental configuration for hands blind and fully blind condition where the user's hands are held under the desk while typing (keyboard visibility absent).



```
the bus was very crowded
the bus was _

          avg    last
WPM      10.25  10.81
ACC      88.8%  72.3%
```

Figure 7: The experimental software showing visual feedback in the normal and hands blind conditions (on–screen feedback present).



```
fine but only in moderation
         _

          avg    last
WPM       26.4  38.64
ACC      95.1% 100.0%
```

Figure 8: The experimental software showing visual feedback in the fully blind condition (on–screen feedback limited).

4.3 Method

The method for the study of blind typing closely resembles the method for the baseline study of novice mini–QWERTY text entry described above (Section 3.1).

4.3.1 Design and Procedure

The blind study is structured as a 3 x 5 within–subjects factorial design. We present the participants three conditions (normal (*n*), hands blind (*hb*) and fully blind (*fb*)) during five sessions which lasted approximately 75 minutes each. The sessions were separated by at least two hours and by no more than two days and scheduled over the course of 9 days. Each session was split into three 20 minute parts delineated by typing condition and separated by five minute breaks. The order of conditions was randomized across participants and sessions. Similar to our previous work, participants were compensated \$0.125 $\times$ wpm $\times$ Accuracy, with a \$4 minimum per condition.

We recruited participants who had completed the first mini–QWERTY keyboard study. Eight of the seventeen original participants self–selected to participate in the blind study. All participants were considered expert mini–QWERTY typists having completed 600 minutes of training prior to beginning the blind study. Four hundred minutes of training came from participating in our previous study. Since there was a delay of about three months between studies, we also had the participants practice for an additional 200 minutes just prior to the commencement of this study. Our participants ranged in ages 18-24. Four participants were female and all were right–handed.

Before the first session, the researcher gave each participant verbal instructions explaining the task and goals of the experiment. The researcher also described the three different typing conditions. The participants were instructed to type as quickly and accurately as possible and to use only their two thumbs to enter text.

As before, each condition began with a warm–up round which consisted of the phrases “abcd efgh ijkl mnop” and “qrst uvwx yz” repeated twice. The warm-up phase was not counted in the statistics. The remainder of the condition consisted of a number of trial blocks containing ten randomly selected phrases. Each participant completed as many blocks as he or she could in the twenty minute period.

4.3.2 Equipment and Software

We continued to use the two mini–QWERTY keyboards (shown in Figure 2) from the baseline evaluation (see Chapter 3 and again employed the Twidor software package. The software was self–administered under researcher supervision. Depending on the condition under test, the software has two different visualization modes. For the normal and hands blind conditions, the program displays the transcribed text as it is entered (Figure 7). For the fully blind condition, the software does not show this feedback. Instead, only a cursor moves across the screen with no characters displayed as the user types (Figure 8). The test software also provides statistical feedback to the participant. We show the typing rate, measured in words per minute (WPM) and the accuracy (ACC) for the most recent sentence typed and the current session average.

4.4 Results

In total, the 8 participants typed 13,920 phrases across all sessions. Typing rate suffered considerably in both visually impaired conditions (Figure 9). In the first session, an ANOVA shows that there is a statistical difference between conditions ($p < 0.05$). A post–hoc analysis shows there is not a statistical difference between the hands blind and the fully blind conditions ($p = 0.820$) while there is a difference between the normal condition and the two blind conditions ($p_{hb} < 0.05$ and $p_{fb} < 0.05$). The normal typing rate ($M_n = 53.99$ wpm, $SD_n = 10.34$) is similar to our previous experiment. In contrast, the typing rates dropped for both blind conditions. The hands blind typing rate started at $M_{hb} = 40.61$ wpm ($SD_{hb} = 11.46$) for the first session and the fully blind typing rate was $M_{fb} = 41.76$ wpm ($SD_{fb} = 8.07$).

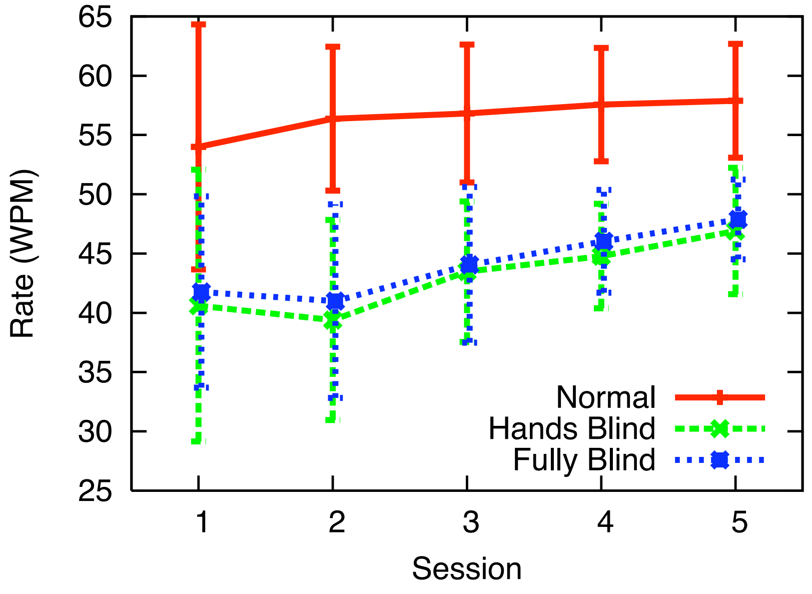


Figure 9: Mean typing rates with ± one standard deviation for the three conditions averaged across keyboards. The Normal condition is red, Hands Blind is green, and Fully Blind is blue.

At the end of the our fifth session, the hands blind typing rate increased to $M_{hb} = 46.90$ wpm ($SD_{hb} = 5.33$) and the fully blind rate to $M_{fb} = 47.88$ wpm ($SD_{fb} = 3.35$). As expected for expert usage, the normal condition did not show a corresponding increase ($M_n = 57.89$ wpm, $SD_n = 4.80$). While the blind rates increased, they are still statistically different from the normal condition ($p_{hb} < 0.001$, $p_{fb} < 0.001$). This performance drop represents a decrease of 11 wpm which is approximately 20% of normal typing speed.

The trends seen in the typing rates also continue in the accuracy data (Figure 10). Typing accuracy was drastically reduced with the introduction of the blind conditions and gradually improved with time. An ANOVA shows statistical difference between conditions ($p < 0.05$). A post–hoc analysis still reveals no statistical difference between the accuracy rates for the hands blind and fully blind conditions ($p = 0.357$) though there remains a difference between the normal condition and the two blind conditions ($p_{hb} < 0.05$ and $p_{fb} < 0.05$). After the initial session, the accuracy rate for the normal condition was $M_n = 92.4\%$ ($SD_n = 5.36\%$) while hands blind was $M_{hb} = 81.9\%$ ($SD_{hb} = 8.57\%$) and the fully blind was $M_{fb} = 77.8\%$ ($SD_{fb} = 8.44\%$).

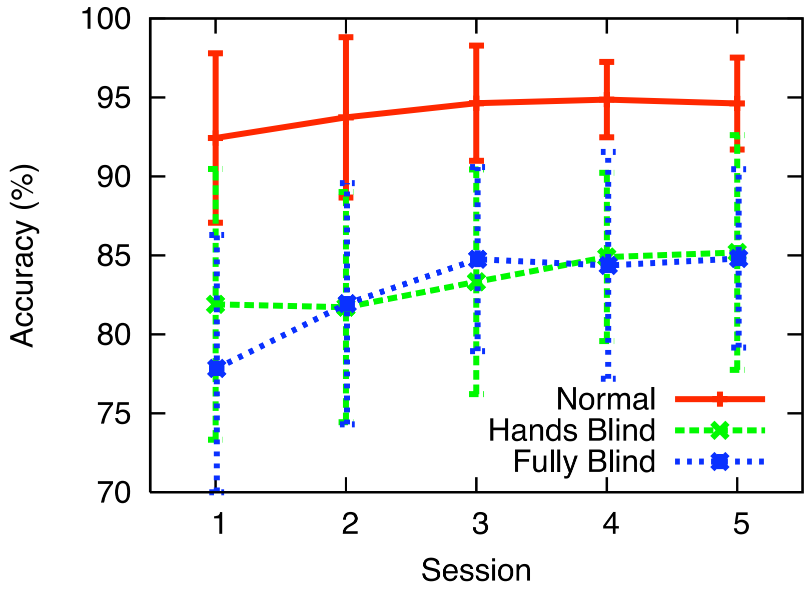


Figure 10: Mean accuracies with ± one standard deviation for the three conditions averaged across keyboards. The Normal condition is red, Hands Blind is green, and Fully Blind is blue.

Examining the accuracy rates at the end of the final session shows that the hands blind typing condition accuracy increased to $M_{hb} = 85.2\%$ ($SD_{hb} = 7.43\%$) and fully blind to $M_{fb} = 84.8\%$ ($SD_{fb} = 5.64\%$). Again, the normal condition did not show a corresponding increase ($M_n = 94.6\%$, $SD_n = 2.91\%$). While the blind accuracy rates increased, similar to the typing rates, they are still statistically different from the normal condition ($p_{hb} < 0.01$, $p_{fb} < 0.01$).

4.5 Blind Discussion

On the whole, the mini–QWERTY keyboard data show that the participants in the blind conditions initially decrease in performance and slowly start to recover. This

Condition	**Session 1**	**Session 5**
Normal	53.99 wpm (SD = 10.34)	57.89 wpm (SD = 4.80)
Hands Blind	40.61 wpm (SD = 11.46)	46.90 wpm (SD = 5.33)
Fully Blind	41.76 wpm (SD = 8.07)	47.88 wpm (SD = 3.35)

Table 6: Mean mini–QWERTY typing rates and standard deviations captured in the normal, hands blind and fully blind conditions for sessions 1 and 5.

drop mirrors Silfverberg's results but contrasts with our past work on the Twiddler wherein there was no performance drop when transitioning to limited visual feedback conditions.

Our mini–QWERTY participants learned to type while looking at the keyboard. Anecdotally, we observed that while typing in the normal condition, participants would read a phrase displayed on the monitor, look down at the keyboard, type the phrase, press enter to submit the phrase, and look back at the monitor to read the next phrase. This pattern of behavior was no longer valid upon introduction of the blind typing conditions to our expert mini–QWERTY users. The blind conditions are sufficiently different that the participants were forced to partially relearn how to type without looking at the keyboard which explains the initial decrease in typing rate and accuracy observed as the participants were, in effect, blind typing novices. As they proceeded through the sessions, they gradually relearned how to type and their performance increased. While their performance did rebound, it is important to reiterate that none of our participants were able to meet or exceed their normal typing rate or accuracy while typing in a blind condition during the experiment.

It is also worth noting that the total time the participants spent typing blind was not equal to the time spent in the normal condition. In effect, the two blind conditions combine to give participants 40 minutes of practice in a state of limited keyboard visibility for every 20 minutes of regular typing (there was no statistically significant difference between the two blind conditions). Therefore, the data for the fifth session do not strictly represent five typing sessions of 20 minutes, but instead a total of 200 minutes of practice in a keyboard visibility absent condition.

Another important issue with our study on blind mini–QWERTY typing relates to our experimental setup. With our study, the participants typed in sustained sessions gaining practice with blind typing. This situation may not be representative of real world conditions where it is unlikely that users would have multiple sustained sessions of practice with limited visual feedback. Instead, most of the user's experience in blind situations would likely be short and intermittent while trying to accomplish some other primary task like taking notes while sitting in a meeting.

CHAPTER V

TEXTING WHILE WALKING: AN EVALUATION OF MINI-QWERTY TEXT INPUT WHILE ON-THE-GO

In this chapter we discuss our study designed to investigate the impact of mobility on individuals' ability to input text on mini-QWERTY keyboards.

The goals of the study outlined in this chapter are

1. To determine the average typing rates and accuracies for expert mini–QWERTY keyboard typists inputing text on mini–QWERTY keyboards while mobile (either walking, sitting, or standing).

2. To make methodological contributions to the emerging field of controlled use–in–motion studies

Interacting with mobile technology while in-motion has become a daily activity for many of us. It is not unusual to observe individuals inputing text on a mobile phone while walking to a destination. Common sense leads one to believe that texting while mobile can be dangerous since users are distracted and not paying attention to the environment. In fact, previous studies have investigated this area and found that mobility does negatively impact text entry performance. However, this research has only focused on novice participants typing on virtual keyboards on touch screen mobile phones. As such, it is unclear if this intuition holds for expert typists or if using keyboards that provide tactile feedback can mitigate the issues that arise when texting and walking.

In this chapter, I investigate the impact of mobility on users' ability to quickly and accurately input text on mobile phones equipped with fixed-key mini-QWERTY keyboards. In total, 36 participants completed 600 minutes of typing on mini-QWERTY keyboards (300 minutes training up to expertise) in three mobility conditions (walking, seated, and standing for an additional 100 minutes each) generating almost 4,000,000 characters across all conditions. In our evaluation of expert texters we found that walking has a significant impact on expert word-per-minute rates but not on their accuracy rates, illuminating that even under ideal conditions (expert typists receiving tactile feedback from their interactions), mobility impairs mobile interaction.

5.1 Introduction

Many individuals carry mobile devices all the time and use these devices in a wide variety of contexts including at work, at home, in the car, while riding public transportation, et cetera [32]. Often, mobile device usage occurs while the user is on-the-go. The most common mobile device in use today is the mobile phone, and interacting with a mobile phone while walking has become common practice. It is not unusual to observe individuals walking while using their mobile phones to make phone calls, browse the web, or read and write emails or text messages. Using a mobile device in this manner can result in distraction since the user is forced to split their attentional resources between their mobile device and the environment [44]. As mobile devices become ever more powerful and portable, they will be used in an increasing variety of situations. Already it is not uncommon for mobile phones to be the first object that people interact with in the morning and the last one with which they interact before going to bed [7].

Kane et al. observed that most mobile interfaces are designed to be used by a person who is standing still and attending solely to the task of interacting with the mobile device [25]. However, the portable nature of these devices results in usage that occurs while the user is often on-the-go. When mobile, users must constantly adapt their usage of a device to the demands of their environment [26]. When walking, a user's head and hands both move making it difficult for the user to read the screen or interact with the mobile phone. For example, when attempting to input text on a mobile phone while walking, a user must maintain awareness of her surroundings while at the same time navigating her environment and interacting with a device that is itself in motion. In these situations, the user's ability to accurately interact with the device is often impaired. Sears et al. used the term *situational impairments* to describe the situation that occurs when contextual factors reduce a user's ability to successfully interact with a mobile device in a way that is similar to how user's with physical or sensory impairments interact with technology [50].

Despite the increasing importance of mobile devices in everyday life, little research has been published that quantifies the impact of mobility on the use of mobile devices. The degree to which that distraction impacts performance is unknown. In this chapter, we describe an investigation that explores the impacts of mobility on interaction with a mobile device. Specifically, we examine the effects of sitting, standing, and walking on users' ability to quickly and accurately input text into a mobile phone equipped with a physical mini-QWERTY keyboard (see Figure 11). We compare words-per-minute and accuracy rates in each condition. We conducted this study using a walking track constructed in our laboratory in order to fully observe the effects of typing on a mini-QWERTY keyboard while walking.

We present a study designed to investigate the impact of mobility on individuals' ability to input text on mini-QWERTY keyboards. Our work makes the following contribution to HCI and text entry research communities: We conduct a large controlled text entry evaluation, we determine the average typing rates and accuracies for expert mini–QWERTY keyboard typists inputing text on mini–QWERTY keyboards while mobile (either walking, sitting, or standing).

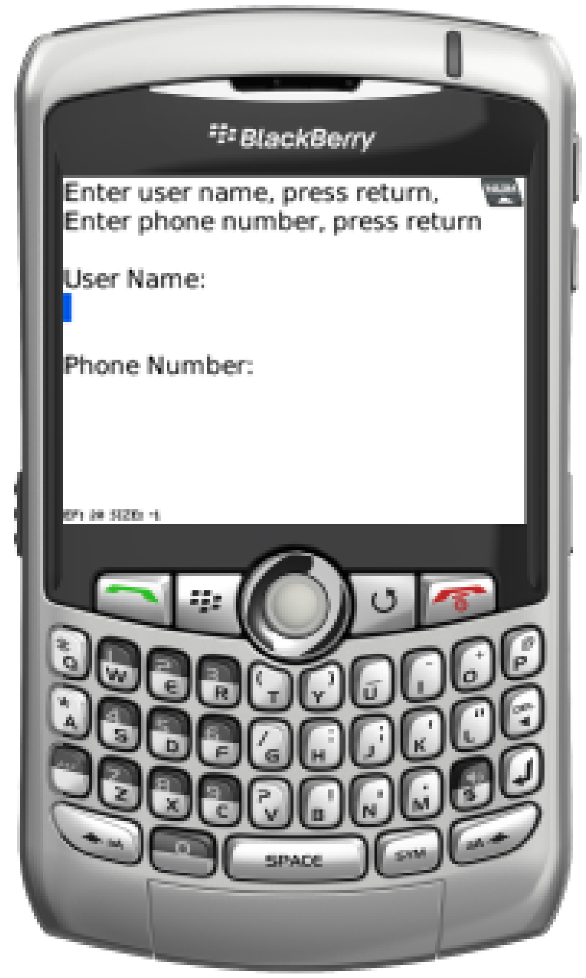

BlackBerry
Enter user name, press return,
Enter phone number, press return
User Name:
Phone Number:
SPACE

Figure 11: The mini–QWERTY keyboard enabled mobile phone used in the mobile study: the RIM Blackberry Curve 8320.

5.2 Evaluation

Though much research has explored the use of mobile devices while in-motion, current studies have not investigated such topics as typing on physical keyboards, mobile errors, or the impact of user expertise on on-the-go mobile input. The goal of our evaluation is to investigate the impact of mobility on expert fixed-key mini-QWERTY keyboard text entry performance.

5.2.1 Participants

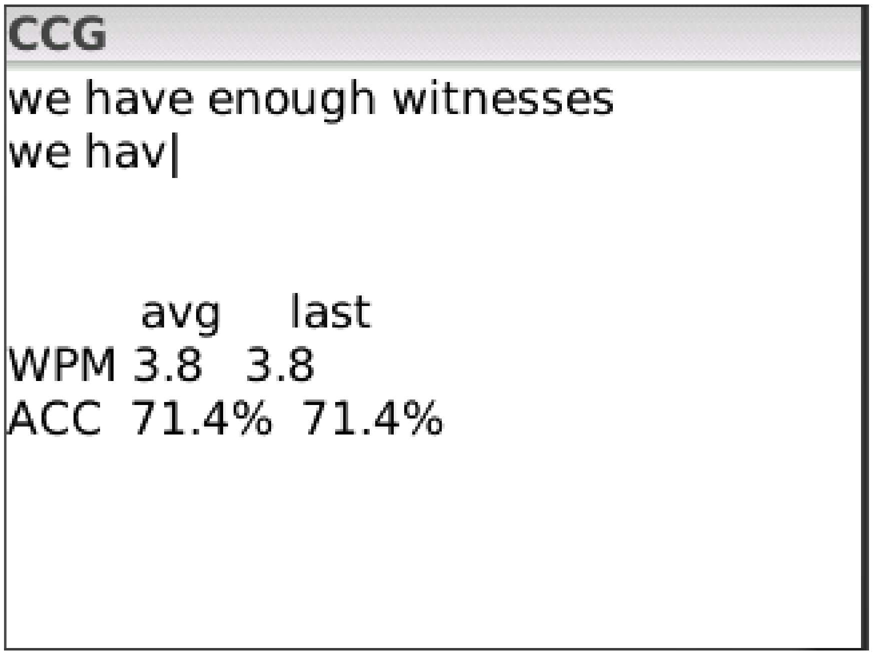
We recruited 36 participants (15 female, 21 male) ranging in age from 18 to 25 (M=19.75, SD=1.63) who had not used a mini–QWERTY keyboard more than once. Each participant had at least six years of full–QWERTY keyboard experience (M=11.36, SD=2.86) and at least one year of mobile phone experience (M=5.11, SD=2.19). All participants were native American English speakers who were taught to read and write in an American English education system (it is important from a spelling/errors perspective that participants were American English natives). On average, participants sent 27 text messages a day (SD=43). Twenty-nine of the thirty six participants reported that they have sent a text message while walking, sending on average 13 text messages a week while walking (SD=16).

5.2.2 Equipment and Software

Figure 11 shows the mini–QWERTY keyboard enabled mobile phone used in the experiment, a RIM Blackberry Curve 8320. For this study, we ported the Twidor testing software to the Blackberry platform resulting in the new application BlackTwidor (though BlackTwidor was run on a RIM Blackberry Curve 8320, it can run on any current Blackberry model mobile phone).

As with our previous studies, the software was configured to use the MacKenzie and Soukoreff phrase set [37], a set of 500 phrases representative of the English language. The test software presented a phrase to the participants at the top of the screen. Underneath the presented phrase, participants transcribed the phrase and pressed the return key upon completing the task. The test software provided statistical feedback in the form of typing rate and accuracy data for the most recent sentence typed and the current session average (see Figure 12).

Each session was preceded by a warm–up phase, in which the participants were asked to type the phrase “abcd efgh ijkl mnop qrst uvwx yz” twice. Participants were



we have enough witnesses
we hav|

```
     avg    last
WPM 3.8   3.8
ACC  71.4%  71.4%
```

Figure 12: The interface in the study displayed both the presented string and the transcribed string (in progress directly below the presented). It also showed the words-per-minute (WPM) and accuracy rates (ACC) for the session so far (avg) and the phrase just completed by the participant (last).

required to type this phrase without mistakes. The warm–up phrase was not counted in the session statistics. Upon completion of the warmup, the twenty-minute session was started. A session was comprised of a number of trial blocks each containing ten randomly selected phrases from the MacKenzie and Soukoreff phrase set [37]. The participants completed as many blocks as possible in a twenty–minute session. At the end of a twenty-minute session the mobile phone displayed a message instructing the participant to find the researcher so that the researcher could collect the data, record the participants' performance, and compensate the participant appropriately.

5.2.3 Procedure

Prior to beginning the first session of the study, the participants filled in a questionnaire detailing their demographic information and their mobile phone usage history. Upon completion of the questionnaire, participants were given an introduction to the task and were provided a mobile phone to familiarize themselves with the application. Participants were instructed to type using only their two thumbs and to type as quickly and accurately as possible. Participants were allowed to interact with the mobile device and were prompted to ask any clarifying questions prior to beginning the task. When the researchers were confident that the participant was comfortable performing the task, they were ushered into the lab wherein they proceeded to input text for the entirety of a twenty-minute session. Sessions were completed in pairs with a 5–minute break after the first 20–minute session. Each session pair was separated by at least two hours and by no more than two days. To complete the study, participants scheduled fifteen session pairs over the course of two to three weeks.

5.2.3.1 The Walking Track

Recently, some use-in-motion studies have started having a researcher walk in front of the participant to set a constant walking pace for the duration of the task [25, 17]. While using a human pacer does help ensure consistency across participants, it has the potential to force participants to commit unnatural errors since perhaps a participant needs to slow down to perform particularly difficult tasks. Though not to the same degree, using a human pacer introduces some of the same external validity challenges as using a treadmill to evaluate use-in-motion. The participants' pace is forced and their ability to make navigational decisions is reduced. Without allowing participants the flexibility to set their own pace and walk their own path we are unable to accurately measure the true impact of mobility on individuals abilities to interact with mobile technology while on-the-go.

We take a different approach and choose to simply quantify natural human performance. When in the walking condition, participants were instructed to walk at a normal pace around a track constructed in our laboratory (see Figure 13). The track was approximately 25.2 meters long and was denoted with pairs of flags hanging from the ceiling with their tips 0.75 meters apart. Each flag was hung so the tip was approximately 1.6 meters above the floor. We chose to use flags hanging from the ceiling to ensure that participants were engaged in a head-up task. Had the participants been following a path laid out on the ground, a head-down condition would have ensued which we considered to be inappropriate given the nature of the study (as walking around, head down, is not typical behavior). As the study is conducted with participants walking continuously around the track, they were told they could slow down or stop as needed to complete a trial, but were asked to keep walking at a comfortable pace if possible.

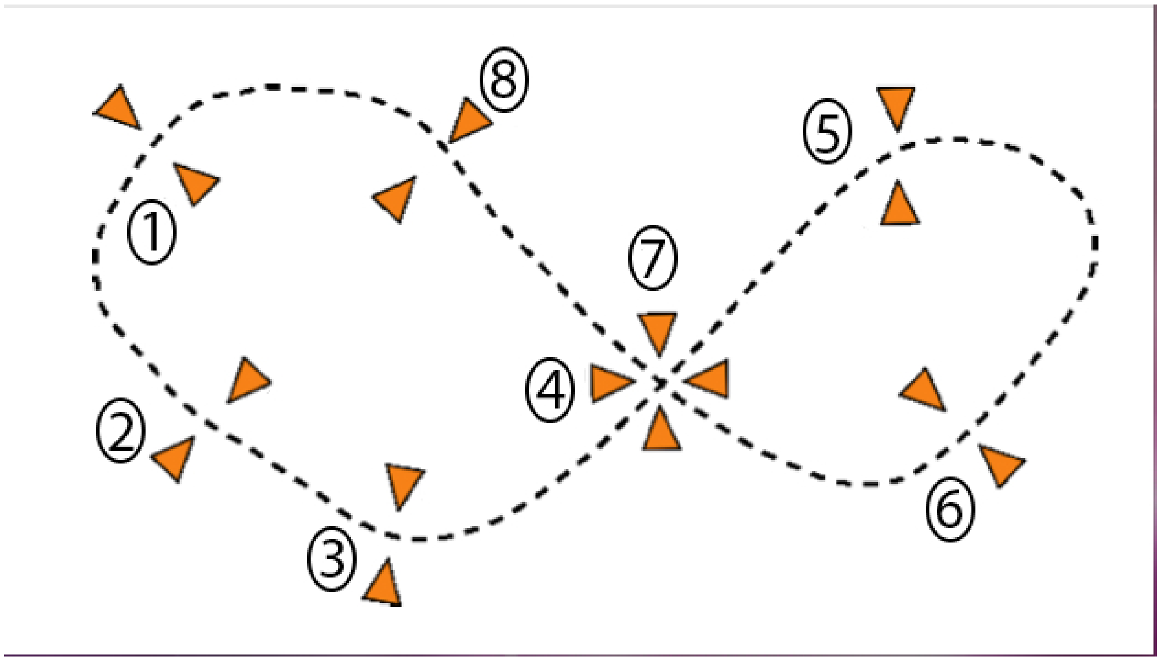


Figure 13: The path participants walked, starting at flag 1 and proceeding either clockwise or counterclockwise.

In an effort to accurately calculate the speed and distance traveled by the participants, we built a set of motion sensors that we mounted in the ceiling of our lab between each pair of flags (see Figure 14). The sensors were connected to a computer in the laboratory via Bluetooth. Every time a participant walked between a pair of flags, the sensor would record the time and send that information to the computer. In this way, we could calculate the instantaneous and average speed of the participant as they walked from flag to flag along the track as well as the total distance traveled (see Figure 15). Upon completion of the twenty-minute session, the mobile phone displayed the following message in a large font on the screen "Session over. Please hand the Blackberry to the researcher." Participants were requested to stop where they stood in order to allow the researcher to measure how far they had walked past the last pair of flags. In this way we were able to accurately record the total distance traveled to the nearest half-meter.

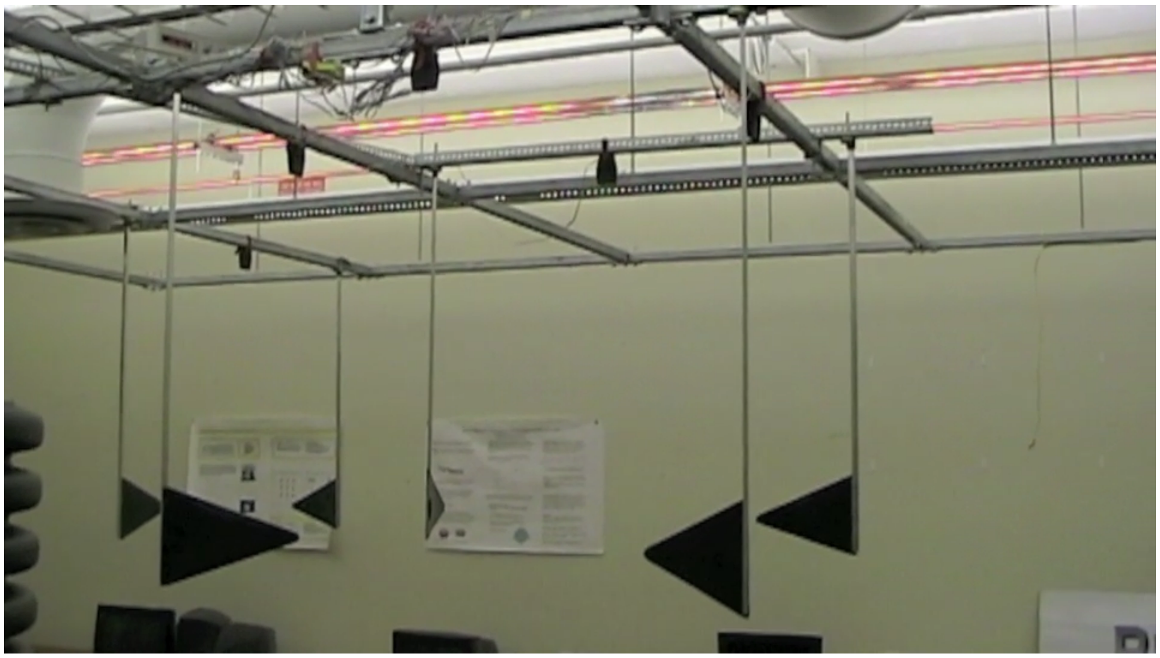


Figure 14: The flag and sensor configuration that comprised the walking track.

5.2.4 Study Design

To begin, participants were trained up to expertise before being introduced to the mobile conditions. Previous studies have shown that participants' learning curves flatten after 300 minutes of typing on a mini-QWERTY keyboard indicating that participants have become expert typist [9]. In this study, participants were asked

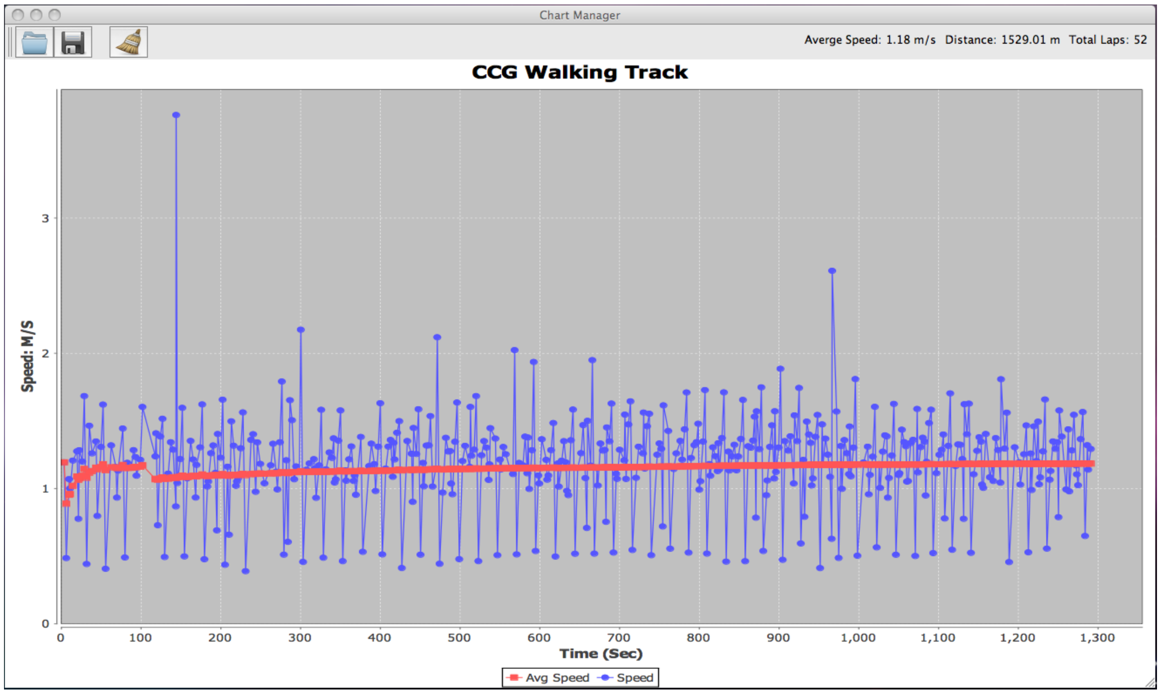


Figure 15: A screenshot of the software we used to capture participant movement data. The y-axis is speed in $\frac{m}{s}$ and the x-axis shows time in seconds. Each blue dot on the chart represents an instance where the participant walked through a pair of flags (under a sensor). The blue dots show instantaneous pace while the red dots show the participant's average speed. Average speed, distance traveled and total number of laps around the track can be seen in the upper right corner.

to complete 15 twenty-minute sessions seated in our laboratory prior to entering the experimental phase of the evaluation. Having completed 300 minutes inputting text on a mini–QWERTY keyboard enabled mobile phone, our now–expert participants transition into the mobility portion of the study.

The study is designed to investigate expert text entry performance in three different mobility conditions: sitting (the control condition), standing (participants stand in an empty room for the duration of the twenty-minute session) and walking (participants continuously walk the track in our laboratory for the duration of a twenty-minute session). Each participant was assigned an initial mobility condition and rotated through all three conditions over the course of the study. Participants input text for five 20-minute sessions (100 minutes) each mobility condition. The order of mobility conditions was counterbalanced and participants were randomly assigned to orders. All 36 expert participants successfully input text for 100 minutes in all three mobility conditions. By the completion of this study, participants had typed for a total of 600 minutes – 300 minutes seated and 300 minutes in the mobile conditions (100 minutes sitting, 100 minutes standing, and 100 minutes walking).

We structured the study as a 3 x 5 within-subjects design. The factors and levels were:

- **Mobility** *Sitting, Standing, Walking*
- **Sessions** *One, Two, Three, Four, Five*

Because our participants were expert typists by the time they began this phase of the evaluation, there was no need to counterbalance conditions at the session level to avoid learning affects. Participants input text for five sessions in each mobility condition before switching to a different condition. To ensure that even if learning effects did present themselves, we would have at least 100 minutes of typing data in all three mobility conditions that were uncompromised. These data from the first 100 minutes of the experimental phase of the evaluation could be then used to compare performance between the three different mobility conditions and we would still have usable data. Analyzing the subset of participants who went directly from the final training session into the Sitting mobility condition (thus typing for 400 minutes while seated) confirmed that our participants' learning curves had flattened. Additionally, running an ANOVA over sessions 16-20 showed no significant effect on either WPM or ACC.

5.3 Results

Thirty-six participants input 131,884 phrases, and 3,872,505 characters across all sessions. 60,818 phrases and 1,793,564 characters were typed in the first three hundred minutes of typing as participants were trained to expertise. In the experimental portion of the study, a total of 24,116 phrases and 705,156 characters were typed while seated, 24,078 phrases and 705,014 characters were typed while standing, and 22,872 phrases and 668,771 characters were typed while walking (see Table 7). Averaged over the five 20-minute walking sessions, participants traveled 912m (SD=300m) or approximately 2.75km/hour.

Study Phase	**Phrases**	**Characters**
Training to Expertise (300 minutes)	60,818	1,793,564
Seated (100 minutes)	24,116	705,156
Standing (100 minutes)	24,078	705,014
Walking (100 minutes)	22,872	668,771
Total (600 minutes)	131,884	3,872,505

Table 7: Dataset collected from 36 participants typing for 600 minutes: 300 minutes seated training from novice to expert, 100 minutes seated, 100 minutes standing, and 100 minutes walking.

The main measures collected were

- speed, calculated as words per minute (see Equation 6)
- accuracy (see Equation 7), and, when participants were mobile,
- distance traveled and
- average speed.

In the mobility conditions, the mean entry rate for Seated was 56.79 WPM (SD=11.51), while Standing was 56.61 WPM (SD=10.97), and Walking was 52.51 WPM (SD=11.56). The mean accuracy for Seated was 95.36% (SD=6.15%), Standing was 95.25% (SD=6.54%), and Walking was 94.91% (SD=6.59%) (see Table 8 for complete results). On average, participants traveled 911 meters (SD=300m) per 20-minute session. The minimum total distance traveled over the course of 100 minutes was 1,298 meters, while the maximum total distance traveled over the course of 100 minutes was 7,247 meters.

Study Phase	**Mean WPM (SD)**	**Mean ACC (SD)**
Training to Expertise (300 mintues)	48.21 (13.11)	96.05% (6.87%)
Seated (100 minutes)	56.79 (11.51)	95.36% (6.15%)
Standing (100 minutes)	56.61 (10.97)	95.25% (6.54%)
Walking (100 minutes)	52.51 (11.56)	94.91% (6.59%)
Total (600 minutes)	52.06 (12.78)	95.58% (6.65%)

Table 8: The average words per minute and accuracies for 36 participants in the training phase of the study and the three mobility conditions (seated, standing, and walking).

5.3.1 Training to expertise

An analysis of variance (ANOVA) of text entry speed shows a main effect for session ($F_{14,490} = 75.158$, $p < 0.0001$). The main effect of session was expected as it is assumed that the participants would learn to type faster over the course of the 15 twenty-minute sessions. We performed the same test for accuracy which also shows a main effect for session ($F_{14,490} = 3.205$, $p < 0.0001$). Again this effect was expected since accuracy rates typically decrease when typing rates increase due to the speed/accuracy tradeoff.

5.3.2 Mobility Conditions

We tested for effects of condition order on the main measure of typing speed using a 3-way ANOVA with mobility condition order as a between-subjects factor and Session and Mobility as within-subjects factors. No main effect of mobility condition order was found, indicating that overall counterbalancing on the condition level was effective.

The 36 participants typed a total of 71,066 phrases and 2,078,941 characters across all sessions and all mobility conditions. We analyzed the data using a mixed model analysis of variance with fixed effects for Mobility and Session. We found a main effect of Mobility on speed ($F_{2,56} = 13.0076$, $p<.0001$) as well as a main effect of Session on speed ($F_{4,112}$=15.905, $p<.0001$). To analyze these effects further, we conducted post-hoc pairwise comparisons using Bonferroni correction. The post-hoc test for Mobility shows a significant effect on speed between Sitting and Walking (t_{35} = 5.244, p<.0001) and a significant effect on speed between Standing and Walking (t_{35} = 4.762, p<.001) but no significant effect on Speed between Sitting and Standing. The post-hoc test for Session shows a significant effect on speed between Sessions 1 and 3 (t_{35} = -5.251, p<.0001), Sessions 1 and 4 (t_{35} = -4.557, p<.0001), Sessions 1 and 5 (t_{35} = -8.585, p<.0001), Sessions 2 and 3 (t_{35} = -5.201, p<.0001), Sessions 2 and 4 (t_{35} = -3.264, p<.002), and Sessions 2 and 5 (t_{35} = -5.518, p<.0001). No other statistically significant effects on speed were found for Session. We found no main effect of either Mobility on accuracy ($F_{2,56}$=1.316, p=.276) or Session on accuracy ($F_{4,112}$=1.469, p=.216).

5.4 Discussion

The goal of this study was to quantify the impact of mobility on text entry performance. To demonstrate the impact of mobility on text entry performance, we designed a "best case scenario" study utilizing expert participants typing in a controlled environment on devices that afforded tactile feedback (fixed-key mini-QWERTY keyboards). Even in these idealized conditions we still saw a significant impact of mobility on participants' text entry rates with mobility inducing a significant negative impact on typing performance. Most surprisingly we did not see a corresponding impact of mobility on participant accuracy rates.

Though interacting with mobile technology while in-motion has become a daily activity, typing on a phone while on-the-go leads to situational impairments that negatively impact typing rates. We evaluated expert two-thumb typing on a fixed-key mini-QWERTY keyboard enabled mobile phone. In our longitudinal evaluation, 36 participants who had minimal mini-QWERTY experience trained for 300 minutes to become expert mini-QWERTY typists and then moved into the mobility phase of the study to type for 100 minutes in each of three conditions: sitting, standing, and walking. Surprisingly, walking has a significant impact on expert word-per-minute rates but not on accuracy rates. These results illuminate that even under ideal conditions (expert typists receiving tactile feedback from their interactions), mobility impairs users' ability to interact with a mobile phone while on-the-go.

CHAPTER VI

AN ANALYSIS OF MINI–QWERTY KEYBOARD TYPING ERRORS IN VARIOUS CONTEXTS OF USE.

In this chapter I discuss my analysis of typographical errors made when inputing text on fixed–key mini–QWERTY keyboards in various visibility and mobility contexts. The goals outlined in this chapter are:

1. To determine the types of errors made when typing on fixed–key mini–QWERTY keyboards.
2. To determine the types of errors made when typing on mini–QWERTY keyboards in various mobility conditions.

I conduct an in-depth analysis of typing errors committed on mini-QWERTY keyboards, introduce off-by-one errors and explore how the off-by-one error type has changed as keyboards have improved, and finally discuss semantic errors and present a new method for identifying semantic errors in text entry experiments. For an in-depth discussion of text entry metrics, please see Section 2.5.

6.1 Errors in mini-QWERTY keyboard typing

Mini–QWERTY keyboard typists typically employ two–thumbs when operating a mobile device. Mini–QWERTY keyboards have the same one–to–one key–to–letter ratio as seen on a full desktop QWERTY keyboard. In order to fit such a large number of keys into the space normally occupied by the twelve keys of a mobile phone keypad, the letter keys need to be very small and densely packed together on the mobile device. It is not uncommon for this design to result in keyboards that contain keys with a surface area of less than 25 mm^2 and inter–key spacings of less than two millimeters. The keys are significantly smaller than a user's thumbs which results in difficulty of use. The user's thumbs occlude visibility of the keys, introducing ambiguity as to which key was actually pressed. Further, Fitts' Law implies that typing accuracy decreases as typing speed increases due to the relationship between target size, movement speed, and accuracy [55]. Taken together, these effects combine to lead to errors in mini–QWERTY keyboard text input where a user's thumb presses multiple keys at the same time (usually pressing the key either to the left or the right of the intended key) or presses the intended key twice.

6.1.1 Analysis of errors from the Baseline study

Errors have long been considered an important source of insight into understanding a user's performance of a task. Grudin performed an analysis of error patterns for full–QWERTY desktop typing in an attempt to understand how complex motor task skills are organized and developed [20]. Like Grudin, I perform a similar analysis of the data from the Baseline study (Chapter 3), segmenting errors into groups: substitutions, insertions, deletions and transpositions.

Substitution errors occur when a character is replaced by a different character within a text string. Insertion errors occur when a character is added, and deletion errors occur when a character is omitted. Transposition errors occur when a participant attempts to type a letter combination or word and exchanges two of the letters (for example, typing *teh* but intending *the*). With full–QWERTY keyboard typing Grudin found substitution errors account for the majority of the errors (62.92%). Errors on mini–QWERTY keyboards are more evenly distributed with substitutions still occurring most frequently (40.2%) followed by insertions (33.2%) and deletions (21.4%). In comparison, there are relatively few transpositions (5.2%).

6.1.1.1 Substitution Errors

Grudin identifies several different types of substitution errors that occur in full–QWERTY keyboard typing including row errors, column errors, and homologous errors [20]. Row errors occur when the intended character is replaced by a character immediately to the left or the right of the intended character. For example, the user intends to press *e* on a QWERTY keyboard but instead types *w* or *r*. Column errors occur when the correct character is replaced by a character immediately above or below of the intended character (the user intends to press *d* but instead types *e* or *x*). Homologous errors occur when the correct character is replaced by the mirror–image character typed by the same finger in the same position on the other hand (the user intends to press *d* but instead types *k*). Similar to Grudin's work, the majority of mini–QWERTY keyboard substitution errors are row errors accounting for 55.3% of the substitution errors and 22.2% of the total errors.

6.1.1.2 Insertion Errors

The prevalence of insertion errors merits an in–depth analysis. Inspired by the number of row errors discovered in the substitution case, I analyzed insertions in a similar manner. I examined insertions that are off by one key to the left or right of the intended key and include in this measure insertions that occur when a user presses the same key twice. The combination of the row and key repeat insertions account for 68.7% of the insertions and 22.8% of the total errors.

6.1.1.3 Off–By–One Errors

I group row substitutions, row insertions ("roll–on" and "roll–off") and key–repeat insertion errors to form the new error category I call "off–by–one". This error type accounts for 45.0% of all of the errors and occurs more often than any other error type (see Table 9). Off–by–one errors consist of insertions and substitutions of letters on the keyboard directly adjacent to the key the user intended to press. Accidental key repeats are insertions where the user unintentionally presses the same key twice (e.g. the user types"cat**t**" when she intended to type "cat"). Many of the remaining insertions result when the user presses an additional key either immediately to the left or to the right of the intended key, and 92% of these off–by–one insertions can be classified as either Roll–On and Roll-Off insertion errors. Roll–On insertions are where the inserted character comes before the intended character (e.g. the user types "ca**r**t" when she intended to type "cat"). Roll–Off insertions which occur when the inserted character comes after the intended character (e.g. the user types "cat**r**" when she intended to type "cat"). Finally off-by-one substitution errors occur when the intended character is replaced by the character immediately to the right or left of the intended character (e.g. the user types "ca**y**" or "ca**r**" when she intended to type "cat"). Off–by–one errors possibly occur because the thumb spans multiple keys. My data imply that when a user is typing, she may accidentally press multiple keys at the same time with a single thumb. This finding has implications on the ergonomic and physical layout design of mini–QWERTY keyboards. It also could be leveraged to produce better automatic error correction algorithms for mobile text entry.

Error Type	**Percentage**
Off–by–one	45.0%
Deletion	21.4%
Substitution	18.0%
Insertion	10.4%
Transposition	5.2%

Table 9: Updated error rates accounting for off–by–one errors.

6.1.2 Analysis of errors from the mobile evaluation

The first 300 minutes of typing in the mobile study (see Chapter 5 for details) mimics the study design of the baseline evaluation (Chapter 3). The data collected in the baseline evaluation was produced from participants typing on mini-QWERTY keyboards designed as attachments for personal digital assistants. A lot of research and design has gone into the development of robust mobile keyboards since that time. As such, conducting a similar analysis of errors on data collected from the mobile evaluation (Chapter 5) provides an interesting basis for comparison. It is interesting to conduct the same analysis as above to not only replicate the earlier findings but also to see if changes in the keyboard hardware have impacted the types of errors committed by participants.

I opted to use the error classifications described by Wobbrock et al. [60] (see Section 2.5) to conduct the analysis of the mobile data. Investigating the training phase of the evaluation in which 36 participants sat and input text in our laboratory for fifteen 20-minute sessions, I found that of the 1,793,564 characters input, 90,710 were errors. Of note, there were far more uncorrected errors than corrected errors in the set (69,721 vs. 20,989 respectively). This finding is unsurprising as participants will often optimize for speed over accuracy in an effort to maximize their earnings from the study. The most common type of error in the dataset was Uncorrected Substitutions which accounted for 34% of the total errors. The second most prevalent error type was Uncorrected Omissions which accounted for almost 26% of the errors. This finding is particularly interesting given that when participants took the time to correct their mistakes, there were significantly fewer Corrected Omissions than either Corrected Substitutions or Corrected Insertions (see Table 10 for a full breakdown of errors that were collected from the first 300 minutes of typing).

6.1.2.1 Substitution Errors

In my analysis of the errors from the baseline study I found that the majority of mini-QWERTY keyboard substitution errors were row errors accounting for 55.3% of the substitution errors and 22.2% of the total errors. When analyzing data from the mobile study I found that row substitution errors again account for the majority of the errors. Surprisingly, the prevalence of these errors has not decreased dramatically

Error Type	**Error Count**	**% of Total Errors**	**% of Total Chars**
Cor No Errors	17,584	n/a	0.98%
Cor Omissions	4,728	5.21%	0.26%
Cor Insertions	5,774	6.37%	0.32%
Cor Substitutions	10,487	11.56%	0.58%
Uncor Omissions	23,339	25.73%	1.30%
Uncor Insertions	15,532	17.12%	0.87%
Uncor Substitutions	30,850	34.01%	1.72%
Uncor No Errors	1,685,270	n/a	93.96%
Total	1,793,564	100.00%	100.00%

Table 10: A breakdown of the errors committed by 36 participants in the training phase of the mobile evaluation. Participants typed for 300 minutes generating a total of 1,793,564 characters and 90,710 errors.

as keyboard hardware has improved over time. From my data, there were 20,460 row substitution errors. These errors account for 49.50% of the substitution errors and 22.56% of the total errors in my dataset.

6.1.2.2 Insertion Errors

In my earlier analysis of errors from the baseline study I found that a combination of row and key repeat insertions accounted for 68.7% of the insertions and 22.8% of the total errors. Adding key repeats to my analysis of the mobile data does not have a large impact as key repeat errors occurred extremely rarely (only 431 key repeats) in the mobile dataset. Including key repeats in my analysis of insertions shows only a marginal increase in the percentages of row insertions to 32.71% of the insertions and 7.68% of the total errors. I believe that mobile phone manufacturers efforts to fix the key repeat issue accounts for the single greatest differences in mobile phone keyboards today compared to keyboards from the early 2000's.

6.1.2.3 Off–By–One Errors

Of note, recent investigations of errors in two handed touch screen typing [17, 43] have confirmed our earlier published findings that off-by-one errors are the most common errors that occur when typing with two thumbs on a mobile phone [11]. I found similar results when analyzing the mobile dataset, classifying 30.24% of the errors as off-by-one errors.

Comparing off-by-one errors between the baseline (Chapter 3) and mobile (Chapter 5) evaluations differences in keyboard ergonomics can be observed. Table 11 illustrates how the reduction of key repeat insertions impacts the overall off-by-one error rate.

Error Type	**% of Error Class**	**% of Total Errors**
Baseline row substitutions	55.3%	22.2%
Mobile row substitutions	49.50%	22.56%
Baseline row insertions	68.7%	22.8%
Mobile row insertions	32.71%	7.68%
Baseline off-by-one errors	N/A	45.0%
Mobile off-by-one errors	N/A	30.24%

Table 11: A comparison of the off-by-one errors committed in the baseline evaluation and the off-by-one errors committed in the training phase of the mobile evaluation. The second column shows the number of row errors divided by the total number of errors for each error class per dataset. The third column shows the number of row errors divided by the total number of errors per dataset.

6.1.3 Mobile Errors: Sitting vs. Standing vs. Walking

To investigate mobile errors I decided to perform an analysis similar to the one I performed when analyzing the data from the first 300 minutes of the evaluation. This analysis resulted in the following breakdown of errors across mobility conditions (see Table 12). To investigate the impact of mobility on error type I performed an ANOVA with fixed effects for Mobility and Error Type. As participants typed more slowly when walking they generated less text in a twenty minute period than they did when sitting or standing. As such, I normalized the error data (errors/character) to account for this discrepancy. I did not include either Corrected NoError or Uncorrected NoError classes in my analysis. As I was uninterested in correction behavior at this point, I combined corrected and uncorrected error classes. As such, my ANOVA utilized Mobility with three levels (Sitting, Standing, and Walking) and Error Type with three levels (Omissions, Insertions, and Substitutions) I found no main effect of Mobility on the number of errors committed ($F_{2,70}$=.390, p=.679). I also did not find an interaction effect of Mobility $*$ Error Type on the number of errors committed ($F_{4,140}$=.123, p=.974). However, I did find a main effect of Error Type on the numbers of errors committed ($F_{2,70}$=30.825, p<.0001). To analyze these effects further, I conducted post-hoc pairwise comparisons and employed Bonferroni correction. Unsurprisingly, the post-hoc test for Error Type showed a significant effect on number of errors committed between Omissions and Substitutions (t_{35} =-7.289, p<.0001) and between Insertions and Substitutions (t_{35} = -6.348, p<.0001) but no significant effect on number of errors committed when comparing Omissions and Insertions (t_{35} =1.711, p=.096).

	Error Count			% of Total Errors			% of Total Chars		
Error Type	**Sitting**	**Standing**	**Walking**	**Sitting**	**Standing**	**Walking**	**Sitting**	**Standing**	**Walking**
Cor No Errors	3,645	3,663	2,461		n/a		0.52%	0.52%	0.37%
Cor Omissions	1,304	1,277	969	3.52%	3.38%	2.58%	0.18%	0.18%	0.14%
Cor Insertions	1,284	1,243	1,036	3.47%	3.29%	2.76%	0.18%	0.18%	0.15%
Cor Substitutions	2,186	2,073	1,827	5.90%	5.49%	4.87%	0.31%	0.29%	0.27%
Uncor Omissions	9,185	9,521	9,988	24.80%	25.22%	26.62%	1.30%	1.35%	1.49%
Uncor Insertions	6,757	7,012	7,074	18.24%	18.57%	18.86%	0.96%	0.99%	1.06%
Uncor Substitutions	16,320	16,631	16,621	44.07%	44.05%	44.30%	2.31%	2.36%	2.49%
Uncor No Errors	664,475	663,594	628,795		n/a		94.23%	94.12%	94.02%

Table 12: A breakdown of the errors committed by 36 participants in the three mobility conditions: sitting, standing, and walking. Participants typed for 100 minutes in each condition generating 705156 characters and 37036 errors while sitting, 705014 characters and 37757 errors while standing, and 668771 characters and 37515 errors while walking for a total of 1,791,169 characters and 145,971 errors.

Next I investigated correction behavior in an effort to understand if correction behavior changed in the different mobility conditions. To investigate correction behavior in the mobility conditions, I again conducted a repeated measures ANOVA with fixed effects for Mobility (Sitting, Standing, Walking), Error Type (Omissions, Insertions, Substitutions), and Correction (Uncorrected, Corrected). Again, I saw no main effect for Mobility or interaction effects for Mobility and Error Type, Mobility and Correction ($F_{2,70}$=.604, p=.549), or Mobility, Error Type, and Correction ($F_{4,140}$=.221, p=.926). I did however see main effects for Error Type and Correction ($F_{1,35}$=44.440, p<.0001) as well as an interaction effect of Error Type and Correction ($F_{2,70}$=18.508, p<.0001). To investigate these effects further, I again conducted a series of post-hoc pairwise comparisons employing Bonferroni correction. Comparing all corrected to all uncorrected errors yielded a statistically significant result (t_{35} =-6.666, p<.001). Exploring the interaction effect between Error Type and Correction yielded a significant effect for number of errors for almost every combination of Error Type and Correction. Of note, no significant effect was found when comparing Corrected Omissions to Corrected Insertions (t_{35} =-.166, p=.869) or Uncorrected Omissions to Uncorrected Insertions (t_{35} =1.813, p=.078) implying that participants' took the same approach to correcting (or not correcting as was more typically the case) Insertion and Omission errors. All other comparisons were statistically significant.

6.1.4 Semantic Errors

In this section I describe my analysis of a type of error that I term "Semantic Errors." A semantic error occurs when a user inserts an entire word or multiple words into the input stream when transcribing text into a mobile device. These errors are likely a result of memory slips that occur when participants are attempting to input text rapidly. I believe that these errors occur most often as participants progress through extended longitudinal text entry evaluations. As the process of rapidly reading a presented string and then inputting that string as quickly and accurately as possible into a mobile device becomes routinized by the participant, I posit that participants "chunk" at the phrase level instead of at the word level. Chunking is a phenomenon wherein an individual breaks a sequence of numbers or letters into memorable "chunks" in an effort to scaffold their ability to remember the entirety of a long sequence of letters or numbers [40]. The process of rapidly reading a presented string and then inputting that string as quickly and accurately as possible into a mobile device affords a surprising number of these types of errors.

Word level insertions can dramatically impact a participants error rate as they end up committing a series of character level errors in the process of inserting a new word. However, I would argue that there is only one error being committed and it is an error at the word level, not at the character level. To clarify this position, consider the following example:

Presented text: every apple from every tree

Input stream: every apple falls from every tree

Transcribed text: every apple **falls** from every tree

In this example, the participant was asked to type "every apple from every tree" but the ended up typing "every apple **falls** from every tree". Analyzing this example applying traditional (Wobbrockian) error metrics would lead the text entry researcher to conclude that the participant committed the errors seen in Table 13.

Using the Wobbrockian error metrics to calculate the accuracy of the phrase results in an accuracy of 81.82% (see Equation 8). This number is calculated as 1-TER using the equation for Total Error Rate (TER) from [53] (see Equation **??**) where C is the number of correct characters in the transcribed string, INF is the number of incorrect and not fixed errors (uncorrected) in the transcribed string, and IF is the number of

Error type	**Error count**
Cor No Errors	0
Cor Omissions	0
Cor Insertions	0
Cor Substitutions	0
Uncor Omissions	0
Uncor Insertions	***6***
Uncor Substitutions	0
Uncor No Errors	27

Table 13: Analyzing the example above, a text entry researcher applying a Wobbrockian analysis of character-level errors would uncover this error behavior. Applying a semantic analysis to the same example however reveals that in fact, the participant committed a single semantic error, not six individual character level errors.

characters that are incorrect but fixed (i.e. characters in the input stream that do not exist in the transcribed text).

$$TER = \frac{INF + IF}{C + INF + IF} = \frac{6 + 0}{27 + 6 + 0} = \frac{6}{33} = .1818... \tag{8}$$

However, I argue that instead of creating 6 Uncorrected Insertions (counted as INFs), the participant simply committed a single Semantic Error and as such could be counted as a single Incorrect and Not Fixed (INF) error resulting in the following accuracy (see Equation 9):

$$1 - TER = \frac{INF + IF}{C + INF + IF} = 1 - \frac{1}{28} = 96.43\% \tag{9}$$

In order to quantify the number of semantic errors in my data I used the following algorithm. Given all presented strings and all transcribed strings from all users, the algorithm will output the minimum number of semantic errors in the dataset. For each phrase, I compute the Wobbrockian error metrics from the transcribed and presented strings. Based on these metrics I searched for sequences of incorrect insertions bound by spaces. Each of these sequences is considered a potential word and therefore a potential semantic error. To confirm that a sequence of incorrect insertions is in fact a semantic error, I check each potential word using a dictionary built from all the words in the MacKenzie and Soukoreff phrase set. The number of potential words found in the dictionary that are spelled correctly is the minimum number of semantic errors in the data. Applying this algorithm to the data from the mobile evaluation I uncovered 470 semantic errors. Upon further analysis I realized that I had one participant who, perhaps out of boredom, typed several phrases similar to the following:

Presented text: yes you are very smart

Input stream: yes you are very smart a a a a a a a a a a a a a a

Transcribed text: yes you are very smart a a a a a a a a a a a a a a

As each instance of “a” counted as a Semantic Error (i.e. 14 in the above example), I removed her phrases from the analysis resulting in a total of 343 Semantic Errors committed in 332 phrases. From the 343 Semantic Errors, the average length of a Semantic Error is 2.5 characters. The most common Semantic Error was “the” which was inserted 91 times followed by “a” which was inserted 63 times. The longest Semantic Error logged was “deserves” (eight characters), followed by “getting” and “tonight” both of which are seven characters long.

It was not uncommon for a particular phrase to have the same Semantic Error inserted in the exact same location by multiple participants. For example,

Presented text: house with new electrical panel

Transcribed text: house with **a** new electrical panel

occurs 16 times in the data and was committed by 10 different participants at least once (two participants committed this exact error three times, and two committed it twice). There are 38 instances of a phrase having the exact same semantic error committed in the same location multiple times. There are some interesting examples wherein participants inserted different Semantic Errors into the same location within a phrase. For example:

Presented text: this library has many books

Transcribed text 1: this library has **too** many books

Transcribed text 2: this library has **so** many books

Transcribed text 3: this library **is filled ith** many books

The third transcribed text highlights one of the limitations of my method. It is obvious through inspection that the participant intended to insert the words "is filled with" into the phrase. However, they omitted the "w" from "with" and as such I only calculated two Semantic Errors instead of the three. As mentioned above, my current analysis of Semantic Errors returns only the minimal number of Semantic Errors.

6.2 Discussion

Though the mobile study showed a significant impact of walking on text entry rates, most surprisingly I did not see a corresponding impact of mobility on participant accuracy rates. Intrigued by this result, I conducted an in-depth analysis of errors that mirrored my earlier analysis of errors from the baseline study. In comparing errors from the training phase of the mobile evaluation to the errors in the baseline study, I uncovered that changes made to keyboard technology have dramatically reduced the number accidental key repeats made by participants. In the experimental phase of the mobile evaluation I investigated the impacts of sitting, standing and walking on the types of errors committed by participants. I uncovered a statistically significant difference between the number of Substitution errors and the number of Insertion and Omissions errors committed. Comparing corrections in the training phase of the mobile study to corrections in the experimental phase of the mobile study indicates that as participants increase in experience, they correct far fewer mistakes than they do as novices. Though I did not incorporate Semantic Errors into my accuracy calculations, I demonstrated not only the existence of Semantic Errors, but also that they artificially negatively impact accuracy rates. As such, I believe Semantic Errors should be considered to be a single error when calculating accuracy or error rates instead of a series of errors.

CHAPTER VII

AUTOMATIC WHITEOUT: LEVERAGING FEATURES OF THE USER'S INPUT TO CORRECT ERRORS IN MINI–QWERTY TYPING

By analyzing features of users' typing, Automatic Whiteout detects and corrects up to 32.37% of the errors made by typists while using a mini–QWERTY (RIM Blackberry style) keyboard. The system targets "off–by–one" errors where the user accidentally presses a key adjacent to the one intended. Using a database of typing from longitudinal tests on two different keyboards in a variety of contexts, we show that the system generalizes well across users, model of keyboard, user expertise, and keyboard visibility conditions. Since a goal of Automatic Whiteout is to embed it in the firmware of mini–QWERTY keyboards, it does not rely on a word level dictionary. This feature enables the system to correct errors mid–word instead of applying a correction after the word has been typed. By correcting errors mid-word, Automatic Whiteout has the potential to correct errors before the user notices that they have committed a mistake thus minimizing user distraction. Though we do not use a dictionary, we do examine the effect of varying levels of language context in the system's ability to detect and correct erroneous keypresses.

In this chapter I introduce and validate the potential of Automatic Whiteout: a

This chapter is an excerpt from James Clawson, Alex Rudnick, Kent Lyons, and Thad Starner, "Automatic Whiteout: Discovery and Correction of Typographical Errors in Mobile Text Input," in *the proceedings of MobileHCI, 2007, [13]* and from James Clawson, Kent Lyons, Alex Rudnick, Robert A. Iannucci Jr., and Thad Starner, "Automatic Whiteout++: Correcting mini–QWERTY Typing Errors Using Keypress Timing,"in *the proceedings of CHI, 2008*[11] machine learning approach to automatically detecting and correcting off-by-one errors. In Chapter 8, informed by the success of Automatic Whiteout, I refine and revise my technique for automatic error detection and correction and introduce FatThumbs. FatThumbs is a simple set of rules that target fundamental aspect of two thumb typing behavior to automatically detect and correct off-by-one errors. Finally, in Chapter 9, I test FatThumbs on live data and evaluate the impact of automatic error correction on mini-QWERTY keyboard text entry performance.

7.1 Introduction

As described in Chapter 6 we previously performed an analysis of errors that occur when people type on mini–QWERTY keyboards and found that off–by–one errors account for approximately 45% of the total errors committed.

Noticing that these types of errors occur frequently inspired us to attempt to employ machine learning techniques to automatically detect and correct off–by–one errors based on features of the users' typing.

7.2 Automatic Whiteout

Automatic Whiteout [13, 11], is able to detect and correct insertion (Roll–on, Roll–off, and key repeats) and substitution errors that occur in mini–QWERTY keyboard typing data. It employs decision trees, a machine learning technique, to detect errors by recognizing patterns in certain features of the user's typing. Having identified an error, the algorithm corrects the error by deleting the errorful character in the case of insertion errors or by replacing the errorful character with the potentially correct character based on letter, bi–letter or tri–letter frequencies in the case of substitution errors.

In this chapter we demonstrate how Automatic Whiteout could be incorporated into a mobile device by showing that it is generalizable. Specifically, we demonstrate that the algorithm can generalize to different levels of user expertise, to different models of keyboards, and to typists inputting text in conditions of limited feedback. Finally, we evaluate the effect of the correction on overall keystroke accuracy and discuss how our algorithm can be employed to improve mobile text input on mini–QWERTY keyboards with the goal of correcting errors before they are noticed by the user.

The current version of Automatic Whiteout incorporates 82 features. Many of these features take advantage of simple language features, specifically bi–letter and tri–letter frequencies. While Automatic Whiteout does not include a dictionary, we do include letter probability tables based on a large plain–text corpus. We have also allowed the algorithm to look at subsequent keypresses (in addition to prior keypresses) when evaluating a potential error. We call the number of additional keys that the algorithm evaluates first- and second-order contexts. In all our tests, we only use symmetrical contexts (e.g. the first order context is one keystroke in the future as well as one keystroke in the past). In the section Generalization Across Corpora, we explore how context improves system performance. In addition to these features, we also use features such as the keys pressed, the timing information between past and subsequent keystrokes around the letter in question, a letter's frequency in English, and the physical relationship between keystrokes (whether the keys involved are located physically adjacent to each other horizontally).

While the decision trees can become quite complex, a simplified example of how Automatic Whiteout classifies Roll–off insertion errors is illustrative (see Figure 16). The Roll–off classifier first determines if the key of the letter it is inspecting (in this case the letter **"Y"**) is located on the keyboard either one key to the left or to the right of the key of the previous letter (in this case, the letter **"T"**). Next it examines if the time between the previous keystroke and the current keystroke is less than or equal to a threshold (in this case 47 milliseconds). In our testing below, this

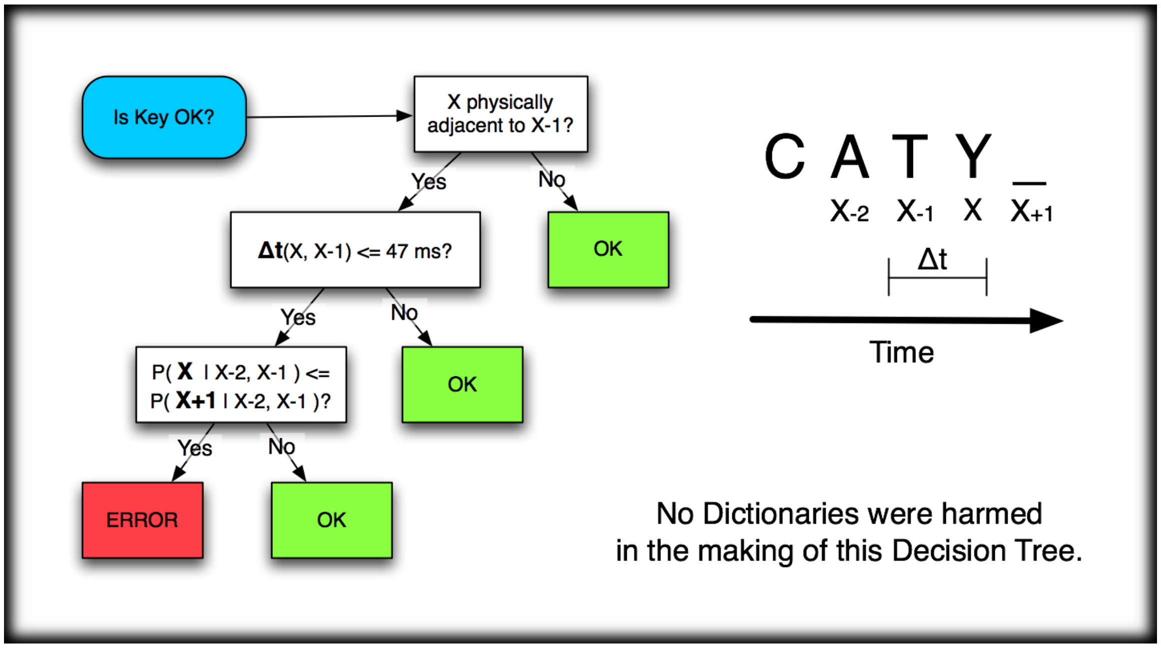

Is Key OK?
X physically adjacent to X-1?
Yes
No
OK
Δt(X, X-1) <= 47 ms?
Yes
No
OK
P(X I X-2, X-1) <= P(X+1 I X-2, X-1)?
Yes
No
ERROR
OK
C A T Y _
X-2 X-1 X X+1
Δt
Time
No Dictionaries were harmed in the making of this Decision Tree.

Figure 16: Example Roll–Off Decision Tree.

timing information is the key to correcting errors without mistakes. Finally, Automatic Whiteout compares the probability of the current key to the probability of the previous key given each key combination's specific tri–letter frequencies and then classifies the current key as a Roll–off insertion error according to these probabilities. In this case, the three letter combination **"CAT"** occurs much more frequently in English that the three letter combination **"CAY"**. Similar trees are learned for detecting the other types of errors as well. The final Automatic Whiteout system tests each keystroke as a key repeat, Roll–on, Roll–off, and substitution error sequentially, stopping the process and correcting the keystroke if any test returns a positive result.

The correction of Roll–on, Roll–off, and key repeat insertion errors is relatively simple. The system deletes the offending key stroke thus removing the insertion. Substitution errors, however, require more information to correct. Letter frequency, bi–letter frequency, and tri–letter frequency are used to help correct off–by–one substitution errors. When Automatic Whiteout determines that a substitution error has happened, it compares the letters to the right and left of the key typed and selects

	Baseline Study	**Blind Study**
Date	Fall 2004	Spring 2005
Participants	14	8
Expertise	Novice	Expert
Sessions	20	5
Conditions	2	6
Phrases Typed	33,947	8393
Keystrokes Typed	1,012,236	249,555
Total Phrases Typed	42,340	
Total Keystrokes Typed	1,261,791	

Table 14: The complete mini-QWERTY datasets for the Baseline and Limited Visibility evaluations.

the most probable one. For example, if the user types **"t h r"** and the system determines that a substitution error has occurred, the possible alternatives are **"t h t"** or **"t h e"**. Since **"the"** is more likely than **"tht"** based on the tri–letter frequencies, Automatic Whiteout replaces the **"r"** with an **"e"**. A similar technique for letter disambiguation was used by Goodman et al. [19] previously.

7.3 Experimental Data

Our dataset is the output of the two longitudinal studies that investigate mini–QWERTY keyboard use detailed in Chapters 3 and 4, (see Table 14). Fourteen participants who had no prior experience with mini-QWERTY keyboard typing participated in the original study [9]. All fourteen participants completed twenty 20–minute typing sessions for a total of 400 minutes of typing. Eight subjects continued to the "blind" study.

In the Baseline Study, the participants typed 33,945 phrases across all sessions, encompassing over 950,000 individual characters. Averaged over both keyboards, participants had a mean first session typing rate of 31.72 wpm. At the end of session twenty (400 minutes of typing) the participants had a mean typing rate of 60.03 wpm. The average accuracy rate (as measured using MacKenzie and Soukoreff's Total Error Metric [53]) for session one was 93.88% and gradually decreased to 91.68% by session

Dataset	Phrases	Keypresses	Errors	OBOs	OBO %
Expert Dell	4,480	64,482	2,988	1,825	61.08%
All Targus	16,407	246,966	8,656	4,983	57.56%
All Dell	15,657	272,230	9,748	6,045	62.01%
Blind Targus	3,266	30,187	2,795	2,072	74.13%
Blind Dell	3,516	29,873	3,287	2,527	76.88%

Table 15: The sampled datasets used for all training and testing of Automatic Whiteout.

twenty.

In the Blind Study, the eight participants typed 8,393 phrases across all sessions for a total of 249,555 individual keypresses. Averaged over both keyboards in the blind mini–QWERTY conditions, our participants had a mean first session typing rate of 38.45 wpm. At the end of session five (200 minutes of typing) the participants had a mean typing rate of 45.85 wpm. The average accuracy rate for session one was 79.95% and gradually increased to 85.60% by session five.

Combining both studies we collected 42,340 phrases and 1,261,791 keypresses. The dataset discussed here is available for public use and can be found at `http://www.cc.gatech.edu/~jamer/mq/data`.

7.3.1 Sampling the Experimental Data

We analyzed the data from all sessions of both datasets and identified each character typed as either correct or as an error. If a phrase contained an error, the characters up to and including the error were kept but the characters that occurred after the initial error were discarded. Truncating the phrase in this manner avoids errors that may have cascaded as an artifact of the data collection. Specifically, Twidor highlights errors as the user enters them. Providing users with visual feedback that indicates when they make mistakes potentially distracts the user, increasing her cognitive load and forcing her to make a decision about whether or not to correct the error. This disruption in the participant's natural behavior potentially effects performance, hence the truncation after the initial error. If the initial error is one of the first two characters in the phrase, the entire phrase is discarded. Additionally, all of the sessions in which participants entered text in the “normal condition” were removed from the blind study and are not used in our analysis. Sampling our dataset reduces the number of phrases and key strokes typed to 30,896 and 449,032 respectively. The sampled set contains 20,879 total errors and 13,401 off–by–one errors.

Repeats	
Feature Name	Description
sameasprev	that the current keypress is the same letter as the previous keypress
prob	frequency of finding the current key preceded by the previous two keys. P(X given X-2, X-1)
prevdt	The dt between the current key and the previous key. dt(X,X-1)
nextnext	X+2 's ascii value
Roll–On	
dt	The time between the previous key and the current key dt(X-1, X)
prevcuradjacent	True if and only if the current key is physically adjacent to the previous key.
neighborprobdiff	P(X given X-2, X-1) - P(bestneighbor given X-2, X-1)
Roll–Off	
futdt	The time between the current key and the next key dt(X, X+1)
curfutadjacent	True if and only if the current key is physically adjacent to the next key typed.
prob	frequency of finding the current key preceded by the previous two keys. P(X given X-2, X-1)
Substitutions	
neighborprobdiff	P(X given X-2, X-1) - P(bestneighbor given X-2, X-1)
prob	frequency of finding the current key preceded by the previous two keys. P(X given X-2, X-1)
neighborprob	the probability of the best neighbor occurring after X-2,X-1
futisletter	True if and only if the next key is in [a-z]

Table 16: The most discriminative features learned by the system for each error classifier, ordered by importance.

	Across Expert–Users		Across Expertise		Across Keyboards		Across Visibility	
	Train	Test	Train	Test	Train	Test	Train	Test
Keyboard	Dell	Dell	Dell	Dell	Dell	Targus	Dell (original)	Targus (blind)
Users	6 of 7	7th	6 of 7	7th	7	7	7	4
	(leave–1–out)	(leave–1– out)	(leave–1– out)	(leave–1– out)				
Sessions	Expert	Expert	All	All	All	All	All	All
	(16–20)	(16–20)	(1–20)	(1–20)	(1–20)	(1–20)	(1–20)	(1–5)

Table 17: Summary of training and independent test sets for each of our experiments detailing keyboard, users, and sessions. All tests are user–independent except for the Dell blind study.

7.3.2 Datasets: Dell complete, Dell expert, Targus complete, and blind

The experimental dataset was further segmented into four sets for training and testing purposes: Dell complete, Dell expert, Targus complete, and blind (see Table 15 for the distribution of data across the various sets). We analyzed the data for all twenty typing sessions for the Dell keyboard (Figure 2 bottom). The complete set of Dell data contain 15,657 phrases, 272,230 keypresses, 9,748 errors, and 6,045 off–by–one errors.

By the time participants began the 16th typing session in the original study they were considered to be expert typists (their learning curves had flattened). We analyzed the data for the last five typing sessions. This subset of the Dell data contain 4,480 phrases, 64,482 keypresses, 2,988 errors, and 1825 off–by–one errors and represents expert usage of a mini–QWERTY keyboard. Of the two keyboards used in the studies, the keys on the Dell keyboard are smaller and are more tightly clustered.

Next we analyzed the data for all twenty typing sessions in the original study for the Targus keyboard (Figure 2 top). The complete set of the Targus data contain 16,407 phrases, 246,966 keypresses, 8,656 errors, and 4,983 off–by–one errors. The Targus keyboard is the larger of the two keyboards. The keys are large, spaced further apart, and are more ovoid than the keys on the Dell keyboard.

The blind dataset consists of data from both the Dell and the Targus keyboards. Four participants per keyboard typed in two different blind conditions for five sessions. The blind conditions have been combined to form one set of data (as there was no statistically significant difference in wpm and accuracy rates in the different conditions). This dataset comprises 200 minutes of typing from eight different participants, four of whom used Dell keyboards, and four of whom used Targus. The blind set of data contains 6360 phrases, 55,642 keypresses, 5874 errors, and 4326 off–by–one errors.

7.3.3 Training

To detect off–by–one errors, we use the Weka [15] J48 algorithm to learn decision trees with metacost to weight strongly against false positives (10X). Weighting so heavily against false positives helps to ensure that Automatic Whiteout minimizes the number of errors that it introduces to the user's typing output. This attribute is important as degrading the user experience is the certainly not a goal of the algorithm.

From the expert Dell data in the original study we randomly assigned 10% of the phrases to be an independent test set and declared the remaining 90% to be the training set. We did not examine the independent test set until all features were selected and the tuning of the algorithm was complete.

From the training set we iteratively built a series of four training subsets, one for each error classifier (Roll–on, Roll–off, repeats, and substitutions). The training subsets were built by sampling from the larger training set; each subset was designed to include positive examples of each error class, a random sampling of negative examples, and a large number of negative examples that previously generated false positives (i.e., likely boundary cases). Due to our desire to avoid incorrectly classifying a correct keystroke as an error, we iteratively constructed these training sets and searched for proper weighting parameters for penalizing false positives until we were satisfied with the classification performance across the training set. For a list of the most discriminative features for each error classifier, see Table 16.

7.4 The Evaluation of Automatic Whiteout

In the following sections, we demonstrate that Automatic Whiteout can successfully generalize across users as well as across different levels of user expertise, different visibility conditions (such as typing while not looking at the keyboard), and different models of keyboards (see Table 17).

Error Type	**Average Corrections (possible)**	**Average Detected**	**Average Wrong Corrections**	**Average OBO error Reduction**
Roll–On	37(57.4)	64.43%	2.9	13.36%
Roll–Off	53(63.9)	83.00%	2.3	19.71%
Repeats	14.7(25.9)	56.91%	0.6	5.30%
Subs	25.4(103.1)	24.24%	3.1	8.50%
AW	120.9(250.3)	48.29%	8.9	46.89%

Table 18: Automatic Whiteout across expert users by training and testing on the expert Dell dataset. Automatic Whiteout performance averaged across seven user–independent tests. On average, users made 260.71 off–by–one errors.

7.4.1 Generalization Across User Expertise

In our preliminary work, we felt that Automatic Whiteout would only be suitable for expert mini-QWERTY typists. Using the new features of Automatic Whiteout allows us to further discriminate between correct and incorrect keystrokes and extend the algorithm to correct errors from less experienced typists.

Using the entire Dell dataset from the original study we tested the ability of Automatic Whiteout to generalize across various levels of user expertise. Again we performed leave–one–out testing. This test yields the rate that Automatic Whiteout will detect and correct off–by–one errors at any level of expertise from complete novices (someone who had never used a mini–QWERTY keyboard before) to expert mini–QWERTY keyboard typists. Table 19 shows the results from these tests which are quite encouraging. Given our subject set (expert desktop keyboard users but novice mini–QWERTY users), Automatic Whiteout could have improved their typing accuracies significantly at all stages of their training. This result suggests that Automatic Whiteout can assist both novice and expert users of such keyboards. It is interesting to note that the percentage of average off–by–one error reduction decreased slightly for Roll–on and Roll–off errors. This result is because the proportion of these errors as compared to total off–by–one errors increases as the user gains experience.

Error Type	**Total Corrections (possible)**	**Average Detected**	**Total Wrong Corrections**	**Average OBO Error Reduction**
Roll–On	762(1034)	73.69%	44	12.16%
Roll–Off	1092(1234)	88.49%	38	17.46%
Repeats	485(649)	74.73%	9	7.97%
Subs	1120(2888)	37.02%	181	14.69%
AW	3136(5805)	54.02%	272	52.20%

Table 19: Automatic Whiteout across expertise by employing leave–one–out user testing. Trained and tested across all sessions of the Dell dataset, Automatic Whiteout performance is averaged across seven user–independent tests.

7.4.2 Generalization Across Keyboards

Using both the entire Dell and Targus datasets from the original study we demonstrate that Automatic Whiteout can successfully generalize across different models of mini–QWERTY keyboards. Though all mini–QWERTY keyboards by definition have the same alphabetic keyboard layout, not all keyboards have the same sized keys or the same inner–key spacings. As such, not all mini–QWERTY keyboards are used in the same manner. Generalizing across different keyboard models demonstrates the applicability of using the Automatic Whiteout solution successfully in mobile devices using different models of mini–QWERTY keyboards.

Perhaps the strongest result in this study (Table 20) is that Automatic Whiteout generalized across keyboards. The system had not been trained on either the Targus keyboard or its users' typing in this dataset. Yet the system still corrected almost half of the off–by–one errors, corresponding to over a quarter of the total errors made.

Comparing Table 20 to Table 19 which were both trained on all the Dell data (both novice and expert) shows that the types of errors detected were similarly successful across both keyboards. However, despite being trained on Dell data, the Targus keyboard had a lower error rate in general and proportionally fewer Roll–on and Roll–off errors than the Dell keyboard (probably due to the larger keys of the Targus). Key repeat errors were more common on the Targus, resulting in key repeats having

Error Type	**Total Corrections (possible)**	**Average Detected**	**Total Wrong Corrections**	**Average OBO Error Reduction**
Roll–On	441(666)	66.22%	29	8.55%
Roll–Off	635(765)	83.01%	25	12.26%
Repeats	717(909)	78.88%	9	14.33%
Subs	796(2383)	32.52%	127	13.00%
AW	2378(4723)	50.35%	190	48.05%

Table 20: Automatic Whiteout across different keyboard models. Automatic Whiteout was trained on the entire Dell set and was tested on the entire Targus dataset from the original experiment.

a larger effect on the total off–by–one error reduction, while Roll–ons and Roll–offs had a lesser effect.

7.4.3 Generalization Across Different Visibility Conditions

To generalize across typing in different visibility conditions, we again used the entire Dell dataset from the original study to train the system. As in the previous section, we test on data from both the Dell and the Targus keyboards. However, for this analysis, we use the data for both keyboards from the blind study to evaluate the effectiveness of Automatic Whiteout on errors from typing in conditions of limited feedback. In addition to performing a user–independent test on the blind Targus data, we also tested on the blind Dell data. In the original experiment there were seven Dell keyboard users. Four of those seven users participated in the blind study. Due to anonymity procedures for human subjects testing, we did not retain the identities of the subjects who continued to the blind study. Thus, we cannot perform a user–independent test as with our other analyses. Instead, training on the entire Dell dataset and testing on the blind Dell dataset can be considered neither a user–dependent test nor a user–independent test.

Table 21 shows the results from these tests. As expected, testing on the blind Dell data performed better than testing on the blind Targus data. In the Targus condition, the system was not trained on the users, the keyboard, or the visibility condition.

Error Type	**Total Corrections (possible)**	**Average Detected**	**Total Wrong Corrections**	**Average OBO Error Reduction**
Dell				
Roll–On	166(252)	65.87%	18	5.90%
Roll–Off	188(213)	88.26%	13	6.99%
Repeats	43(70)	61.43%	6	1.49%
Subs	581(1941)	28.75%	37	20.63%
AW	881(2476)	35.58%	74	34.95%
Targus (User Independent)				
Roll–On	68(114)	59.65%	8	2.95%
Roll–Off	138(169)	81.66%	1	6.69%
Repeats	71(92)	77.17%	1	3.38%
Subs	415(1650)	24.06%	37	17.37%
AW	627(2025)	30.96%	47	30.32%

Table 21: Automatic Whiteout across different visibility conditions. Automatic Whiteout was trained on the entire Dell set and was tested on the blind Dell as well as the blind Targus datasets.

Yet it still corrected 30.3% of the off–by–one errors. Arguably, in practice these rates would be higher because one could train Automatic Whiteout on a representative sample of keyboards and operating conditions. Thus, the 22.5% total error corrected in this condition might be considered a low value.

7.4.4 Generalization Across Corpora

Up until this point, the results have been calculated using the letter frequency tables derived from the MacKenzie and Soukoreff phrase set [37]. The phrase set correlates with written English at the single letter frequency level at 95%. However, Automatic Whiteout uses bi–gram and tri–gram letter frequencies to assist in error detection and correction.

Table 22 shows the impact that various amounts of context have on the ability of Automatic Whiteout to successfully identify and correct errors in mini–QWERTY keyboard text input. With no context Automatic Whiteout is able to identify and correct 25.85% of all off–by–one errors. Being able to examine character pairs, Automatic Whiteout is able to identify and correct 35.97% of all off–by–one errors. Three letter context improves the efficacy of Automatic Whiteout to over 50% (50.32%). Using a dictionary does not improve the solution as recognition rates drop slightly from 50.32% to 50.18%. This lack of improved performance when using a dictionary is worth noting — Automatic Whiteout is equally successful using a dictionary as it is without a dictionary. The ability to implement Automatic Whiteout without having to rely on a dictionary enables the solution to be built directly into the firmware of the keyboard rather than being built into the software of the mobile device. As such, the speed performance gained means that the solution has the potential to detect the error and display the correction without interrupting the user. We hypothesize that the ability to detect and correct errors without visually distracting the user (making a correction within milliseconds before the character is displayed on the screen), will enable faster rates of input and generally a better user experience. Additionally, the ability to implement Automatic Whiteout in the firmware of a mobile device enables it to work in concert with error correction software native to a particular mobile device. Automatic Whiteout simply would pass already corrected input to the software correction system which could then proceed as it would normally on unaltered text.

In general, using a dictionary does not improve the results above the use of tri–letter frequencies. However, there is a distinct improvement in results between the use of single letter frequencies and bi–letter frequencies, and the use of bi–letter frequencies and tri–letter frequencies. The only exceptions are the Roll–off errors, which have a consistently high detection rate across language contexts. Given our features, this result suggests that detection of Roll–off errors are most dependent on keypress timings.

Next we perform a sensitivity analysis using different letter frequency data. We generated up to tri–letter frequencies from the Wikipedia database downloaded on

	No Context	**1^{st} Order**	**1^{st} + 2^{nd}**	**1^{st} + 2^{nd} + Dict**
Roll–On	25.12%	43.03%	64.43%	65.42%
Roll–Off	73.38%	79.42%	83.00%	81.43%
Repeats	6.70%	24.31%	56.91%	59.89%
Subs	3.46%	10.66%	24.24%	23.55%
AW	25.85%	35.97%	50.32%	50.18%

Table 22: The averaged results (% of off–by–one errors corrected) of leave–one–out user training and testing on the expert Dell dataset from the original study using different levels of context.

Error Type	**Average Corrections (possible)**	**Average Detected**	**Average Wrong Corrections**	**Average OBO error Reduction**
Roll–On	34.0(57.4)	59.20%	3.1	12.71%
Roll–Off	54.6(64,7)	84.33%	2.7	21.35%
Repeats	11.6(23.3)	49.69%	0.7	4.49%
Subs	14.1(97.7)	14.47%	2.3	4.85%
AW	114.3(243.1)	47.01%	8.9	43.36%

Table 23: Automatic Whiteout across expert users by training and testing on the expert Dell dataset with Wikipedia letter frequencies. Comparing with Table 18, there was a 3.5% absolute reduction in OBO errors corrected.

August 5th, 2007. We processed the data keeping only plain text and removing all punctuation, tags, markups, tables, etc. Table 23 shows the results of using the Wikipedia letter frequencies on the expert Dell dataset. Comparing these results to those of Table 18 shows average off–by–one error reduction decreases by approximately 3% (46.89% vs. 43.36%). This finding gives a more realistic estimate of how the algorithm would perform on more generalized text.

7.4.5 Sensitivity to Timing

Realizing the important role of timing to the success of our solution, we embarked upon an analysis to determine the impact of imprecise clocks on the performance of Automatic Whiteout. This analysis was done in an effort to understand how robust the algorithm is to permutations in the timing information it receives from the keyboard. We artificially reduced the resolution of the timing by rounding to the nearest 5, 10, 50, 100, 500, and 1000 milliseconds. The impact on the number of

Timing	Total False Positives	Total True Positives	Total False Negatives
Pure	15	2,763	2,078
5	19	2,739	2,102
10	23	2,726	2,111
50	45	2,637	2,174
100	43	2,535	2,151
500	3,790	2,596	2,290
1,000	5,924	2,587	2,352

Table 24: A timing sensitivity analysis for Automatic Whiteout across expert users by training and testing on the expert Dell dataset with Wikipedia letter frequencies (timing data reported in milliseconds).

false positives detected by the system can be seen in Table 24. If values are reported to the system with a resolution of at least 100 milliseconds, relatively successful performance of the algorithm can be maintained. If the granularity of the timing information becomes any larger however, the accuracy of the algorithm suffers and would negatively affect the user. Small permutations in the precision of the timing information, however, do not appear to have a large negative impact on performance.

7.5 Conclusion

In general, Automatic Whiteout can correct approximately 25% of the total errors in the dataset (1-3% of the keystrokes typed across users, keyboards, and keyboard and screen visibility conditions). The system introduces less than one tenth as many new errors as it corrects. These false positives could be further reduced with tuning, satisfying our initial concern of the system becoming too intrusive to use. These results are surprisingly good, especially given Automatic Whiteout uses only tri–letter frequencies instead of dictionaries for error detection and correction.

Table 25 provides a summary of the results from this study. While all conditions yielded approximately a 25% total error reduction, the percentage of keystrokes corrected ranged between 1% (in the Targus condition) and 3% (in the Dell blind condition). This result is explained by the distribution of errors made in the different

Test Across	**%Off–by–one Errors Corrected**	**Total Errors Corrected**	**Keystrokes Corrected**
Expert Users	46.89%	28.64%	1.31%
Expertise	52.20%	32.37%	1.15%
Keyboards	48.05%	27.66%	0.96%
Dell Blind	34.95%	26.87%	2.95%
Targus Blind	30.32%	22.48%	2.08%

Table 25: The results of generalizing Automatic Whiteout to different expert users, to users of differing levels of expertise, to different keyboards, across visibility conditions and across visibility conditions and keyboards.

conditions. As Targus users gained experience, they made approximately 25% fewer errors than Dell typists. Meanwhile, in the blind conditions, users doubled their error rates on both keyboards. Using these observations and Table 25 as a guide, Automatic Whiteout would seem to be most effective on smaller keyboards where device visibility is limited. With consumers buying smaller devices and users' desire to "multitask" sending mobile e-mail in a variety of social situations, Automatic Whiteout seems well suited to assist mini–QWERTY typists. If, as we suspect, error correction is time consuming and errors cascade after the first error is made, Automatic Whiteout may not only improve accuracies but also improve text entry rates.

While we are encouraged by our results, many questions remain. Leveraging features of the user's typing and using Automatic Whiteout enables us to detect and correct errors as the user types, often mid–word. As a result, the correction can happen almost transparently to the user, and errors can be fixed before the incorrect character distracts the user. We believe that such automatic keystroke level correction might allow the user to sustain rapid typing speeds since the user will be able to input text without being distracted by errors. In Chapter 8 we discuss further revisions and simplifications made to the automatic error detection and correction algorithm and in Chapter 9 we describe a user evaluation that assesses individuals' reaction to our system and collects mini–QWERTY typing speeds and accuracies both with and without the use of the automatic error correction system. A longitudinal study that gathers live data will allow us to determine the effects, and noticeability, of the system on users. Do users notice the automatic corrections or the false positives? In an informal pilot test conducted in preparation for the longitudinal study, users did not perceive the presence of the correction system when it was active. However, they did commit fewer errors. Unfortunately, the study was not long enough to determine any effect on typing speeds, nor did it reveal whether users might become dependent upon Automatic Whiteout with long–term use. Expert typists often intuitively "feel" when they make an error in typing and anticipate it, pressing delete before visually confirming the error on the screen. How will automatic error correction effect this behavior? Will expert users pause and verify the impact of their preemptive corrections? Such questions certainly merit further investigation.

CHAPTER VIII

FATTHUMBS: A SET OF SIMPLE RULES USED TO DETECT AND CORRECT OFF-BY-ONE ERRORS

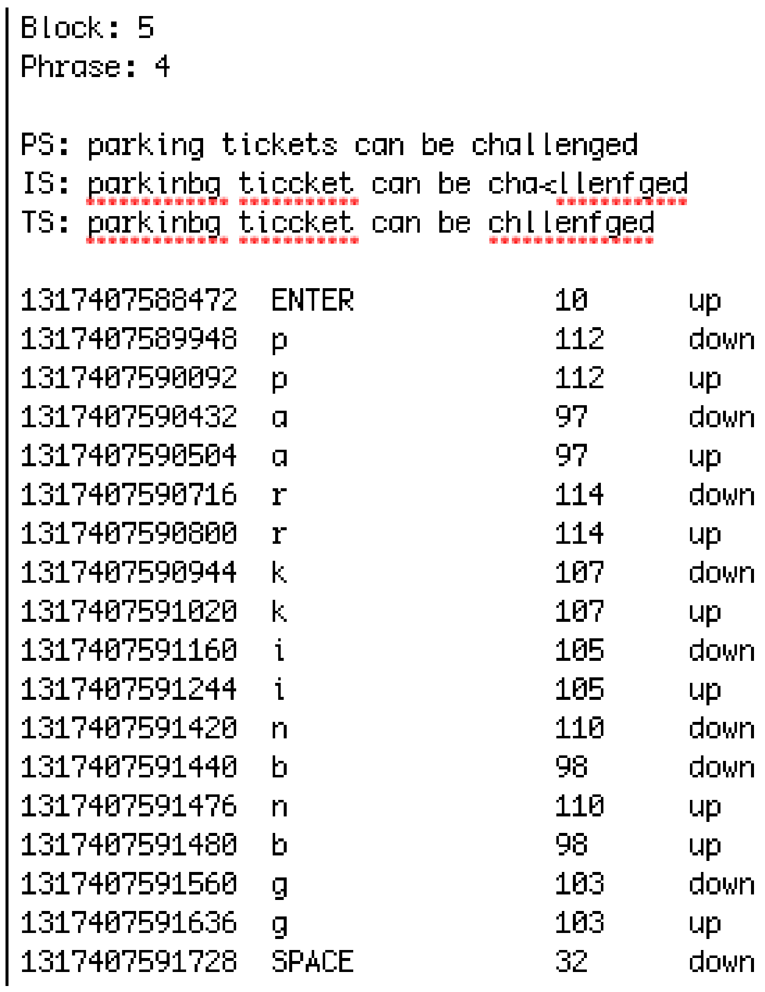
In preparation for the user evaluation of Automatic Whiteout we undertook an effort to optimize the algorithm for Blackberry devices. This process led to an extensive redesign and rethinking of our entire automatic error detection and correction approach. Though heavily influenced by the work in the previous chapter (see Chapter 7), the final version of our error detection and correction solution, the version we used in the user evaluation (see Chapter 9), differs dramatically from Automatic Whiteout to the point that we renamed our approach FatThumbs in an effort to differentiate this work from our previous efforts. The same basic principles used to design Automatic Whiteout guide the design of FatThumbs. Like Automatic Whiteout we still leverage features of the users typing to automatically detect and correct errors. Unlike Automatic Whiteout however, we propose a dramatically simpler solution that not only detects and corrects off-by-one errors but also addresses a new type of error that was discovered from a shift in our data collection method.

The evolution of the algorithm from Automatic Whiteout to FatThumbs began by training the Automatic Whiteout algorithm over an updated set of data collected from the mobile study (see Chapter 5). The mobile data had a higher level of timing fidelity than any of our previous data and was data collected from the phones we intended to use in our evaluation (the Blackberry Curve). We followed this training process with a detailed investigation of the decision trees which resulted in a set of pruned trees that were dramatically smaller than those we had previously considered. Curious to see if the error patterns we had discovered earlier were present in the data collected from the mobile study, we used these decision trees as a visualization tool to inspect our newest data. The trees showed that our error patterns had not greatly changed, though the actual error count differed. We confirmed our earlier insights that off-by-one errors can be primarily detected when individuals' press two keys more quickly than intentionally possible or press two keys that are contextually improbable. The decision trees also revealed a new error type which was not present in the Automatic Whiteout algorithm: two keys pressed at exactly the same time according to the timestamps assigned by the Blackberry. The final step in the evolution from Automatic Whiteout to FatThumbs was replacing our previous solution, an algorithm that stitched together a set of complex decision trees, with a set of simple rules. In effort to create the simplest automatic error detection and correction solution possible, we created FatThumbs, a set of basic rules for detecting off-by-one errors that were crafted with the goal of distilling each error class to its basic components.

8.1 Training on improved data

The mobile evaluation (see Chapter 5) was the first user study that we conducted using Blackberry mobile phones. We leveraged the fact that we were developing new data collection software for the Blackberries, to revise our method of logging keypress data. In our previous studies, we recorded keypress events and timestamps to our log files. With the mobile study we shifted from simply recording keypresses (key-down events) to recording both key-down and key-up events. See Figure 17 for a sample log file.

In Chapter 7.4 we demonstrated that our approach works the best when training our algorithm on one set of data and testing it on the same data, as is expected with machine learning problems. Out of curiousity, we decided to test our existing trees over the data from the mobile study to see how they performed. While this produced decent results, we realized that we had the potential to produce even better results



```
Block: 5
Phrase: 4

PS: parking tickets can be challenged
IS: parkinbg ticcket can be cha<llenfged
TS: parkinbg ticcket can be chllenfged

1317407588472  ENTER            10      up
1317407589948  p                112     down
1317407590092  p                112     up
1317407590432  a                97      down
1317407590504  a                97      up
1317407590716  r                114     down
1317407590800  r                114     up
1317407590944  k                107     down
1317407591020  k                107     up
1317407591160  i                105     down
1317407591244  i                105     up
1317407591420  n                110     down
1317407591440  b                98      down
1317407591476  n                110     up
1317407591480  b                98      up
1317407591560  g                103     down
1317407591636  g                103     up
1317407591728  SPACE            32      down
```

Figure 17: A portion of a log file from the mobile study. The log file shows block and phrase labels, the presented string (PS), input stream (IS), transcribed string (TS), a milisecond timestamp, the key pressed, the corresponding ascii keycode, and the key-up/key-down event respectively.

if we designed new features that could take advantage of our new timing data.

8.2 New features that utilize key-up and key-down information

Given that we now had key-up and key-down events in our logs, we chose to design a whole new set of features that we termed our "automatic timing" feature set. These features were designed to examine changes in the timing between keypresses as well as to examine changes in keypress duration. See Table 26 for a generalized set of our automatic timing features.

Feature Name	Feature Description
dt_dd	the time between two keydowns
dt_du	the time between a keydown and a keyup
dt_uu	the time between two keyups
dt_ud	the time between a keyup and a keydown
dur	the duration of a keypress

Table 26: Keystroke timing feature set that takes advantage of our key-up and key-down logging strategy.

Given that we were always considering a four character set of keystrokes (examining the current keydown and keyup, the future keydown and keyup, as well as the keydowns and keyups of the previous and double previous keystrokes), our automatic timing features examined every combination of these eight events. Each feature is named to describe its function. For example, dt_dd_1,0 is a feature that examines the time between the keydown of the future keystroke and the keydown of the current keystroke (dt = delta in time, dd = down of number before the comma and down for the number after the comma, 1 is the future keypress, 0 is the current keypress) whereas dt_dd_0,-2 is a feature that calculates the difference in downtimes between the current keystroke and the keydown of the keystroke two in the past.

Once we had established these simple building blocks to handle our keypress timings, we could create more interesting and creative features from these basic features. For example, we created two interesting sets of features: one we termed "averages" and the other we named "subtractions." For the "averages" and "subtractions" features, please see Table 27. The "averages" features essentially average two or three of our keystroke timing features and subtract a final feature. The "subtractions" are features where we simply subtract one feature from another. Both of these feature sets give us insight into participants' momentum and pace as they progress through typing a phrase. The motivation behind creating these sets of features was to assess if changes in typing velocity (characters typed per second) could indicate the presence of off-by-one errors.

To demonstrate how these features are calculated, we will walk through an example using Equation 10 to compute the feature “average_du_1” for the “k” entered in Figure 17.

$$average\_du\_1 = (\frac{dt\_du(-2,-3), dt\_du(-1,-2), dt\_du(1,0)}{3}) - dt\_du(0,-1) \qquad (10)$$

As seen in Table 26, “dt_du” is “the time between a keydown and a keyup.” Therefore “dt_du(-2,-3)” would be the time between the down of the “a” minus the uptime of the “p” or 1317407590432 - 1317407590092 = 340ms. We calculate the other “dt_du” values similarly as follows:

$average\_du\_1 = (\frac{dt\_du(-2,-3),dt\_du(-1,-2),dt\_du(1,0)}{3}) - dt\_du(0,-1)$
$= (\frac{340+(1317407590716-1317407590504)+(1317407591160-1317407591020)}{3}) - (1317407590944-1317407590800)$
$= (\frac{340+212+140}{3}) - 144 = (\frac{692}{3}) - 144 = 230.66 - 144 = 86.66$

In this example, average_du_1 = 86.66 ms. This example is just one of the many automatic timing features that we calculate for each valid 4-keypress context. In total, we trained new decision trees using a 104 features, many of which were designed to take advantage of our timing information. For a list of all features, see Appendix A.

Feature Name	Feature Calculation	Feature Description
average_du_1	dt_du(-2,-3),dt_du(-1,-2),dt_du(1,0),dt_du(0, -1)	
average_ud_1	dt_ud(-2,-3),dt_ud(-1,-2),dt_ud(1,0), dt_ud(0, -1)	These features average the first three
average_uu_1	dt_uu(-2,-3),dt_uu(-1,-2),dt_uu(1,0), dt_uu(0, -1)	features and subtract the final one.
average_dd_1	dt_dd(-2,-3),dt_dd(-1,-2),dt_dd(1,0), dt_dd(0, -1)	
average_du_2	dt_du(-2,-3),dt_du(-1,-2)), dt_du(0, -1))	
average_du_3	dt_du(-1,-2),dt_du(1,0), dt_du(0, -1)	
average_ud_2	dt_ud(-2,-3),dt_ud(-1,-2), dt_ud(0, -1)	
average_ud_3	dt_ud(-1,-2),dt_ud(1,0), dt_ud(0, -1)	These features average the first two features
average_uu_2	dt_uu(-2,-3),dt_uu(-1,-2), dt_uu(0, -1)	and subtract the final one.
average_uu_3	dt_uu(-1,-2),dt_uu(1,0), dt_uu(0, -1)	
average_dd_2	dt_dd(-2,-3),dt_dd(-1,-2), dt_dd(0, -1)	
average_dd_3	dt_dd(-1,-2),dt_dd(1,0), dt_dd(0, -1)	
du_sub1	dt_du(1,0), dt_du(0,-1)	
du_sub2	dt_du(-1,-2), dt_du(0,-1)	
ud_sub1	dt_ud(1,0), dt_ud(0,-1)	
ud_sub2	dt_ud(-1,-2), dt_ud(0,-1)	These features subtract the first feature
uu_sub1	dt_uu(1,0), dt_uu(0,-1)	from the second.
uu_sub2	dt_uu(-1,-2), dt_uu(0,-1)	
dd_sub1	dt_dd(1,0), dt_dd(0,-1)	
dd_sub2	dt_dd(-1,-2), dt_dd(0,-1)	

Table 27: Keystroke timing feature set that takes advantage of our key-up and key-down logging strategy.

8.3 Training new decision trees

Using the above features and data from the mobile study, we employed a machine learning software package (Rapidminer) to create new decision trees. Rapidminer wraps the popular machine learning software Weka [15] but adds support for managing very large datasets. To detect off–by–one errors, we used Weka's j48 algorithm (Java implementations of the classic C4.5 decision tree) to create decision trees employing metacost to weight strongly against false positives. We experimented with a variety of weighting schemes and eventually found that weighting 4X against false positives was optimal. This weighting against false positives helps to ensure that FatThumbs minimizes the errors it introduces into the user's typing output.

Using data from the mobile study we randomly assigned 10% of the phrases to be an independent test set and declared the remaining 90% to be the training set. We did not examine the independent test set until all features were selected and the tuning of the algorithm was complete.

From the training set we iteratively built a set of trees by sampling from the larger training set; each training set was designed to include every positive example of each error class, a random sampling of negative examples, and a large number of negative examples that previously generated false positives (i.e., likely boundary cases). We used all errors in the set to train with but only half of the error-free keypresses were ever used as our vast quantity of data overwhelmed the machine we used to build our trees.

To prune the decision trees, we again experimented with many options before arriving at our final implementation. We set our confidence threshold for pruning at 0.5 (0.25 is the default value) and perhaps most importantly chose our minimum instances per leaf to be 100. This means that 100 examples need to be present for a decision to be made and therefore a leaf added to the tree. In practice, varying the minimum number of instances per leaf is a way to dramatically impact the size of a decision tree. For example, when the minimum number of instances per leaf was set to ten, it was common to have trees generated by Rapidminer with more than 200 leaves. Predictably, setting the minimum number of instances to 100 produces trees with approximately 20 leaves. The performance of these trees varies dramatically as trees with a much larger minimum number of instances per leaf will not have the same power as trees that have the flexibility to make more liberal decisions. For an example of a large decision tree (30 instances per leaf) see Appendix B.

Decision Tree Parameters	Non-errors	Substitutes	Roll–on	Roll–off	Non-OBOs	Repeats	Multiple OBOs	# of Leaves	Tree size
8x metacost m10, c.5, 20 % nonerrors	99.89%	47.15%	74.06%	85.84%	34.41%	75.03%	53.24%	277	452
8x metacost m20, c.5, 20 % nonerrors	99.91%	40.37%	63.96%	78.39%	29.23%	66.10%	31.12%	140	229
8x metacost m20, c.5, 50 % nonerrors	99.99%	39.42%	39.54%	83.14%	31.18%	70.33%	27.88%	117	208
4x metacost m25, c.5, 50 % nonerrors	99.92%	44.28%	71.72%	86.41%	32.17%	69.12%	13.85%	111	196
4x metacost m30, c.5, 40 % nonerrors	99.91%	45.16%	72.99%	87.20%	32.65%	64.78%	6.83%	91	156
8x metacost m100, c.5, 20% nonerrors	99.95%	38.43%	46.38%	77.63%	26.47%	33.05%	0%	23	45
8x metacost m100, c.5, 50% nonerrors	99.97%	38.44%	39.69%	78.23%	26.95%	38.12%	0%	22	43
4x metacost m100, c.5, 50% nonerrors	99.96%	40.65%	46.75%	82.09%	27.59%	38.12%	0%	22	43
4x metacost m100, c.5, 50 %nonerrors, no future keyup, 10 features	99.96%	31.53%	43.34%	73.56%	27.41%	0%	0%	21	41

Table 28: Results from several iterations of training and testing decision trees on the data from the mobile evaluation.

Our process of varying the minimum number of instances per leaf revealed the classic machine learning tension between overfitting the data and overgeneralization. We ran tens of iterations, varying decision tree parameters, to train and test decision trees on our data. Each iteration generated ten trees and set of results for how the trees would perform detecting and correcting each error type. As such, we inspected hundreds of trees in an effort to uncover fundamental patterns of user behavior. Table 28 shows the results of several of these iterations. One of the challenges of determining a final set of parameters for our decision trees was balancing between the size of the tree and the performance on each error type. After several iterations, we identified a set of the most common features. To decrease the time it takes to make a decision on each set of four keystrokes we chose to not wait to receive a keyup event on the future keypress. The final set of results in Table 28 shows the performance of decision trees built using just these key features without waiting for a keyup event on the future keystroke. Since these results did not show a significant decrease in performance, we felt confident proceeding with the approach of reducing decision trees to a simple set of rules.

8.3.1 Sample decision tree

Below is a sample J48 decision tree weighted using metacost for 4X against false positives, trained with all errors in the set and half of the non-errors. The confidence threshold for pruning is set at 0.5, and we specified the minimum number of instances per leaf at 100.

```
   W-J48
J48 pruned tree
------------------
prob ≤ 0
| prevcuradjacent_nom = False
| | curfutadjacent_nom = False
| | | dropprobdiff1abs ≤ 0.000934:  repeat (104.0/45.0)
| | | dropprobdiff1abs > 0.000934
| | | | futneighborprob ≤ 0:  nonobo (552.0/11.0)
| | | | futneighborprob > 0
| | | | | neighborprob ≤ 0.011765:  nonobo (143.0/21.0)
| | | | | neighborprob > 0.011765
| | | | | | ud_sub1 ≤ -140:  nonobo (172.0/80.0)
| | | | | | ud_sub1 > -140:  obosubstitute (527.0/109.0)
| | curfutadjacent_nom = True:  rollon (323.0/73.0)
| prevcuradjacent_nom = True:  rolloff (555.0/46.0)
prob > 0
| dt_ud_0_p1 ≤ 124
| | dropprobdiffsign ≤ -1:  rolloff (155.0/36.0)
| | dropprobdiffsign > -1:  nonerror (111.0/49.0)
| dt_ud_0_p1 > 124
| | dt_ud_1_0 ≤ 144
| | | dropprobgain ≤ 0.006861:  nonerror (522.0/62.0)
| | | dropprobgain > 0.006861
| | | | neighborprob1diff ≤ -0.016393:  nonerror (185.0/92.0)
| | | | neighborprob1diff > -0.016393:  rollon (381.0/126.0)
| | dt_ud_1_0 > 144
| | | futprob ≤ 0
| | | | curfutadjacent_nom = False
| | | | | futneighborprob ≤ 0
| | | | | | futnowdiff ≤ -176:  nonobo (141.0/54.0)
| | | | | | futnowdiff > -176:  nonerror (894.0/201.0)
| | | | | futneighborprob > 0
| | | | | | neighborprob ≤ 0.003185:  nonerror (142.0/35.0)
| | | | | | neighborprob > 0.003185
| | | | | | | average_dd_2 ≤ -88
| | | | | | | | letterfreq ≤ 0.04853:  obosubstitute (106.0/35.0)
| | | | | | | | letterfreq > 0.04853:  nonobo (142.0/69.0)
| | | | | | | average_dd_2 > -88
| | | | | | | | neighborprobdiff ≤ 0.009091:  nonerror (181.0/65.0)
| | | | | | | | neighborprobdiff > 0.009091:  obosubstitute (229.0/82.0)
| | | | curfutadjacent_nom = True:  nonerror (129.0/62.0)
| | | futprob > 0:  nonerror (94376.0/1715.0)
Number of Leaves :  21
Size of the tree :  41
```

This example is one of the smaller trees created in our training process. Since we were using these trees as a way to visualize our data, I will now demonstrate how to read the trees, by walking through this example tree to show how we find repeat off-by-one errors. Starting at the top of the tree we have a feature named "Prob." "Prob," is a value that looks up the probability that the current character exists in our dictionary[1] given that the double previous character and the previous character have already been pressed.

If that value is less than or equal 0 then we progress down the tree to check if the previous character and the current character are adjacent. If they are not, we continue down the tree and check to see if the current and the future characters are adjacent. If that value returns false as well, we progress further down the tree. "dropprobdiff1abs" is the absolute value of the result of the p(previous|double previous) - p(current|double previous). If this value is less than a particular threshold value (in this case 0.000934) then the tree returns an off-by-one repeat. One can progress through the tree in a similar manner to understand how Roll-on, Roll-off, and off-by-one substitution errors are determined as well.

We used this tree and many others like this tree to inspect the data from the mobile study. The trees confirmed our understanding that off-by-one errors are still primarily a result of an individual either pressing two keys more quickly than intentionally possible or pressing two keys that are contextually improbable. These trees helped us develop a much deeper understanding of how and why users commit off-by-one errors, and it was through this understanding that we developed our simplified set of rules that became FatThumbs.

8.4 A set of simple rules

FatThumbs is comprised of a set of very simple rules for each error class. These rules typically consist of a timing threshold (determined empirically from our data), the

[1]We built a dictionary of uni-gram, bi-gram, and tri-gram letter probabilities from a set of one million sentences sampled at random from the English version of Wikipedia on February 2, 2012.

location of the keypresses, and a language context rule. Two of the original off-by-one error classes are not included in FatThumbs, and we have added a new error class that was exposed by our decision trees. We are no longer detecting and correcting off-by-one substitution errors. Though these errors are prevalent in the data (they are the largest class of errors in the mobile dataset), correcting off-by-one substitutions simply introduced too many false positives into the input stream. Unlike our other error types whereby if we have a false positive we are deleting a correct character, a false positive substitution actually inserts an incorrect character into the input stream. As having an unexpected character appear on screen is potentially more distracting than missing a character the user thought she had entered, we feel that substitution false positives are more distracting to the user than incorrect deletions. As we were not able to reduce the amount of false positives to the point where we were comfortable including off-by-one substitutions in FatThumbs, we reserve this task as future work. We also chose not to include key repeats as part of the FatThumbs solution. Unlike our earlier datasets which contained a large number of key repeats, the mobile dataset has relatively few key repeats (less than 500 in approximately 1.8 million keypresses). I suspect that the lack of key repeats is due in large part to improvements in the industrial design of mobile keyboards. Essentially, Blackberry has successfully addressed the issue of multiple key de-bouncing. As such, we chose not to focus our efforts on detecting errors that occurred so infrequently.

However, we did uncover a new type of off-by-one error of which we were previously unfamiliar. Equal downtime errors occur when two keys are pressed at exactly the same time, that is both keys have exactly the same (to the millisecond) key-down time. These errors are neither Roll-on errors (which occur when the user rolls onto the correct character from the errorful character) nor Roll-off errors (which occur when the users' thumb rolls off of the intended key striking an adjacent key accidentally) since the keys are pressed at exactly the same time. As such, these errors merited special consideration and their own rules since correction is non-obvious given that the two keys are literally pressed at the same time. Of note, we were unable to detect these errors previously as the hardware used in earlier studies did not sample the keyboard accurately [2].

8.4.1 Detecting Roll-off Errors

A Roll-off error is detected and corrected using the following logic: given the current keystroke and the previous keystroke, if the time between key-up of the current keystroke and key-down of the previous keystroke is less than 170 milliseconds and the two keys are adjacent then an off-by-one Roll-off error has occurred. To correct an off-by-one Roll-off error we delete the current character.

Below is an example of the detection and correction of a Roll-off error taken from the log files collected in Chapter 9. As you can see in Figure 18, the user enters an off-by-one Roll-off error when she accidentally rolls off of the correct character **"u"** onto the incorrect character **"i"**. As FatThumbs is constantly running over a set of four keystrokes, as soon as the user presses **"r"** FatThumbs checks to see if this set of four characters contains an error. In this case, we have an off-by-one Roll-off error on the current keypress (FatThumbs always checks the four character set of "double previous," previous, current, and future keystrokes) where **"i"** is the current character. The time between the key-up time of the **"i"** (524095343ms) and the key-down time of the **"u"** (524095242ms) is 101ms, which is less than the 170 ms threshold. Since the two characters are adjacent on the keyboard, FatThumbs returns a Roll-off error and proceeds to delete the current character (in this case the **"i"**).

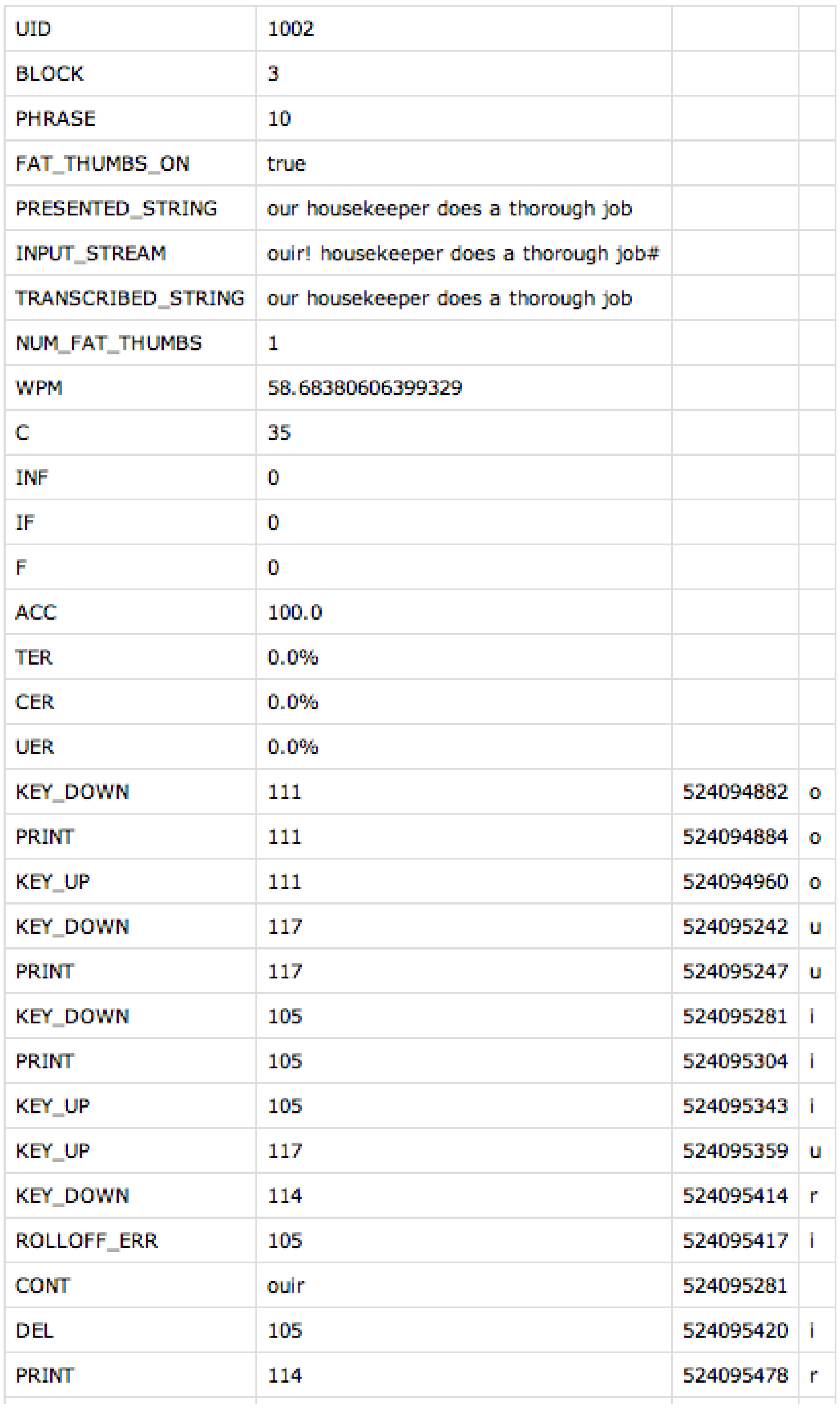
[2]In fact, the phones used in the mobile study (Blackberry Curves) only sampled the keyboard for keypresses every four milliseconds and the timing method we were using artificially added noise to the timestamps, ensuring each key had a unique timestamp. It was only when piloting FatThumbs on the phones we used for the FatThumbs evaluations (see Chapter 9) that we realized the true limitations of our older hardware.

8.4.2 Detecting Roll-on Errors

FatThumbs detects and corrects a Roll-off error using the following logic: given that the future and current keypresses are adjacent, if the time between the key-down of the future and the key-down time of the current is less than 80 millisecond and if the probability of the current given the previous two characters minus the probability of the future given the previous two characters is less than -0.66666666 then an off-by-one Roll-on error has occurred. To correct an off-by-one Roll-on error we delete the current character.

8.4.3 Detecting Equal Downtime Errors

If two characters are adjacent and have the exact same key-down time, then an equal downtime error has occurred. Correcting an equal downtime error, however, is more difficult than simply deleting the current character as there is no way of determining the order of keypresses. To account for this challenge, we check to see if the language context can help determine which character to delete. We employ the same language context rule to the four character set of keypresses as used when detecting Roll-on errors. If the probability of the current character given the previous two characters minus the probability of the future character given the previous two characters is less than -0.66666666 then we delete the current character. If not, then we delete the previous character.



UID	1002		
BLOCK	3		
PHRASE	10		
FAT_THUMBS_ON	true		
PRESENTED_STRING	our housekeeper does a thorough job		
INPUT_STREAM	ouir! housekeeper does a thorough job#		
TRANSCRIBED_STRING	our housekeeper does a thorough job		
NUM_FAT_THUMBS	1		
WPM	58.68380606399329		
C	35		
INF	0		
IF	0		
F	0		
ACC	100.0		
TER	0.0%		
CER	0.0%		
UER	0.0%		
KEY_DOWN	111	524094882	o
PRINT	111	524094884	o
KEY_UP	111	524094960	o
KEY_DOWN	117	524095242	u
PRINT	117	524095247	u
KEY_DOWN	105	524095281	i
PRINT	105	524095304	i
KEY_UP	105	524095343	i
KEY_UP	117	524095359	u
KEY_DOWN	114	524095414	r
ROLLOFF_ERR	105	524095417	i
CONT	ouir	524095281	
DEL	105	524095420	i
PRINT	114	524095478	r

Figure 18: The user commits an off-by-one Roll-off error on the **"i"** of **"ouir"**. FatThumbs succeffully detects the error and deletes the **"i"**. The **"!"** in the input stream indicates an instance of FatThumbs triggering.

CHAPTER IX

FATTHUMBS EVALUATION

In this chapter we discuss our evaluation designed to investigate the impact of automatic error detection and correction on typing performance.

9.1 Evaluation

9.1.1 Equipment and Software

The equipment and software employed in the FatThumbs evaluation was a clear advancement over what used in the Mobility study (see Section 5.2.2). We updated the hardware, moving from the Blackberry Curve 8320 (see Figure 11) to the Blackberry Bold 9900 (see Figure 19). We also augmented BlackTwidor to run FatThumbs.

Due to technical constraints of the Blackberry platform, we were forced to implement FatThumbs in a relatively naive fashion. As we did not have access to the firmware, we implemented FatThumbs in Java. This decision introduced several challenges. Most notably FatThumbs was unable to make a decision on a set of keypresses before the errorful characters were displayed on the screen. Running FatThumbs in the firmware of a phone would eliminate this confound as we would be able to quickly intercept characters after they were pressed ensuring that we display only the correct character (not the error as well) every time a user committed an off-by-one error (and we corrected it correctly). Unfortunately, we were forced print every keypress to the screen. If an off-by-one error was detected, then the errorful character was deleted from the screen. We deleted the errorful character as quickly as possible, but the fact remains that our implementation of FatThumbs had an distinct possibility of distracting the user since we were actively deleting characters from the screen instead of intercepting errors before they were displayed.

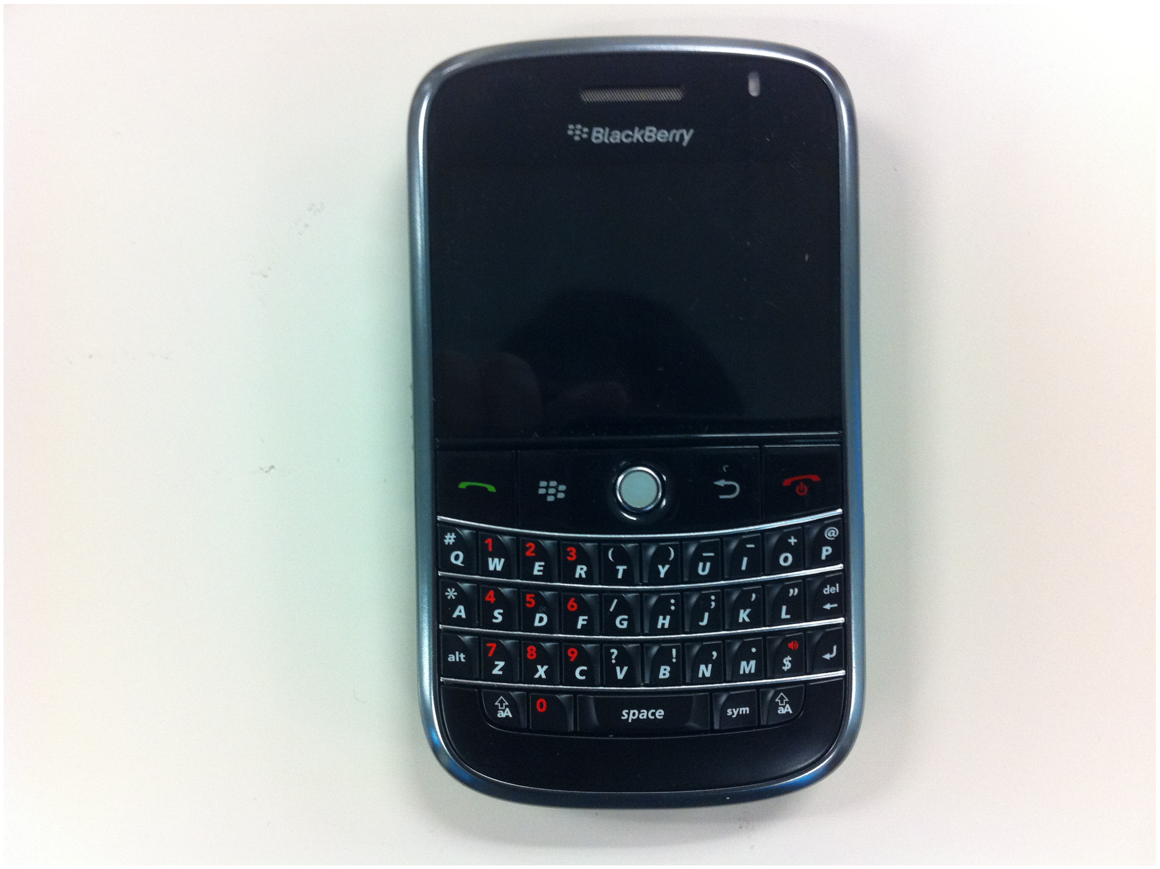


Figure 19: RIM Blackberry Bold 9900: The mini–QWERTY keyboard enabled mobile phone used in the FatThumbs evaluation.

9.1.2 Participants

We recruited 9 participants (2 female) ranging in age from 19 to 23 (M=21, SD=1.36) who had not used a mini–QWERTY keyboard more than once. Seven of the nine participants were right handed. Each participant had at least a decade of full–QWERTY keyboard experience (M=11.33, SD=2.35) and at least five years of mobile phone experience (M=6.00, SD=1.58). All participants were native American English speakers who were taught to read and write in an American English education system. On average, participants stated that they send 21 text messages a day (SD=35). None of the participants had used a device with a mini-QWERTY keyboard more than once.

9.2 Study Design

This study is designed to investigate the performance of our error correction algorithm FatThumbs. As with the mobile study, we trained participants from novice to expertise over the course of fifteen 20-minutes sessions. At the end of 300 minutes of typing, participants entered the experimental phase of the study. In the experimental phase of the study participants completed two 10-block sessions. In each session, FatThumbs was randomly activated at the block level five times. As such, upon completion of the two 10-block sessions, participants had entered 100 phrases with FatThumbs turned on and 100 phrases without FatThumbs [1].

The evaluation took place in our laboratory and participants were seated for the duration of the study. As before, sessions were completed in pairs with a 5–minute break after the first session. Session pairs were separated by at least two hours and by no more than two days. Prior to beginning each session, participants input the warm–up phrase ("abcd efgh ijkl mnop qrst uvwx yz") twice. Participants had to complete the warmup without making any mistakes (or if they did make mistakes, they had to correct the mistakes before the software would allow them to begin a session). The warm–up phrase was not counted in the session statistics. The participants were instructed to type using only their two thumbs and to type as quickly and accurately

[1]In the experimental phase of the study, though participants completed two 10-block sessions, results are only computed over one 8-block session and one 10-block session. The first experimental session had technical challenges that resulted in an unequal balance between the number of blocks participants typed with the algorithm turned on and the number blocks they typed with the algorithm off. Four of the nine participants experienced the technical error which resulted in them typing 6 blocks with the algorithm on only 4 blocks with the algorithm off. The rest of the participants typed 5 blocks on and 5 blocks off. As such, to ensure a balanced dataset, I removed blocks from the first experimental session for each participant to balance the data at 4 blocks on and 4 blocks off for all participants. At the completion of the two sessions, participants had typed 90 phrases with FatThumbs turned on and 90 phrases without FatThumbs (forty in each condition from the first experimental session and fifty in each condition from the second experimental session). The exception is participant 1002 who completed the second experimental session with no mini-SD card inserted into the phone. I was able to record his WPM and ACC measures for the session but his data was lost. As such, only the 8 blocks of typing from the first experimental session are included in the statistical analysis.

as possible. The test software (BlackTwidor with FatThumbs) provided statistical feedback in the form of typing rate and accuracy data for the most recent sentence typed and the current session average.

In addition to the mini–QWERTY data, we also collected demographic data and conducted a brief semistructured interview upon completion of the study.

This study was conducted using deception. Participants were never told that we were evaluating automatic error correction software. From the time they enrolled in the study until the end of the interview, they were only ever told that we were investigating their typing performance over time. It is only at the end of the interview that we revealed to the participants that we were in fact investigating FatThumbs (Appendix C presents the text shown to the participants informing them of the real reason for the evaluation) . We did this in order to be able to assess both how noticeable FatThumbs is as well as how distracting it is to have errors automatically detected and corrected.

9.3 Results

Our nine participants input a total of 16,838 phrases and 517,433 characters over the course of the fifteen 20-minute sessions. In the two experimental sessions participants entered a total of 1,520 phrases and 46,411 characters. In the first 300 minutes of the study, participants committed 14,829 errors of which 3,399 can be classified as off-by-one errors. In the experimental phase of the study, participants committed 261 OBO errors with FatThumbs turned off and 301 OBO errors with the algorithm turned on (see Table 29 for a full description of the dataset).

The main measures collected in this evaluation were words-per-minute (WPM) and accuracy (ACC) (see Section 3.1.2 for details on how both measures are calculated). Over the first 300 minutes of the evaluation when participants were training from novice to expert, the mean entry rate was 46.78 WPM (SD=7.24). The mean

Study Phase	Phrases	Characters	Errors	OBO Errors
Training to Expertise (300 mintues)	16,838	517,433	14,829	3,399
Experimental without FatThumbs	760	23,234	1, 044	261
Experimental with FatThumbs	760	23,177	1,177	301
Total	18,358	563,844	17,050	3,961

Table 29: Dataset collected from nine participants typing for 300 minutes seated while training from novice to expert as well as the data from the experimental phase of the evaluation.

accuracy was 97.43% (SD=2.58%). Over the 760 phrases typed with FatThumbs off the mean entry rate was 56.51 WPM (SD=11.00) while the mean accuracy was 96.82% (SD=2.54%). Over the 760 phrases typed with FatThumbs on the mean entry rate was 58.16 WPM (SD=12.50) while the mean accuracy was 96.21% (SD=2.92%) including correction. See Table 30 for complete results.

Study Phase	WPM (SD)	ACC (SD)
Training to Expertise (300 mintues)	46.78 WPM (SD=7.24)	97.43% (SD=2.58%)
Experimental without FatThumbs	56.51 WPM (SD=11.00)	96.82% (SD=2.54%)
Experimental with FatThumbs	58.16 WPM (SD=12.50)	96.21% (SD=2.92%)

Table 30: Word-per-minute (WPM) and accuracy (ACC) rates for 9 participants typing for 300 minutes as they trained to become expert mini-QWERTY typists. Additionally, we include WPM and ACC rates for the 760 phrases typed with FatThumbs off and for the 760 phrases typed with FatThumbs on.

We ran a two-tailed paired samples t-tests to compare WPM and ACC rates between the two experimental conditions. We did not achieve statistically significant results on accuracy ($t_8 = 1.093$, p=.306). The test comparing WPM with FatThumbs on to WPM with FatThumbs off we did achieve a statistically significant result ($t_8 =$ -2.452, p<.040). These results suggest that when FatThumbs is active, participants type faster than when FatThumbs is off. We were hoping that participants would have typed more accurately as well, however, with faster typing rates came an increase in errors. Fortunately, FatThumbs was able to successfully detect and correct a large portion of those errors which is perhaps one reason why accuracy rates are so similar. The next section presents an in-depth analysis of the errors committed in these two conditions in hopes of illuminating the impact that automatic error detection and correction has on participant performance.

9.3.1 FatThumbs performance

In order to assess the performance of the FatThumbs algorithm I first investigate participant typing behavior and establish a ground truth label for each input character. After determining participant ground truth, I determine the total possible impact of FatThumbs through the identification of all off-by-one errors in the dataset. Having classified all off-by-one errors, I then investigate FatThumbs performance measuring the number of times the algorithm is triggered, the number of times it correctly corrects an off-by-one error, the number of times it misses an off-by-one error and the number of times it incorrectly identifies a character as an off-by-one error resulting in a false positive.

9.3.1.1 Participant ground truth

Though we determined participant ground truth for each character pressed in the study, this section and the following sections will focus only on describing how well FatThumbs performed in the experimental phase of the study when the algorithm was turned on. To assess FatThumbs performance, we first had to establish ground truth regarding participant typing behavior. To do so, we followed the string alignment error detection and classification procedure outlined by Wobbrock et al. in [60]. This procedure yielded labels for each keystroke in the dataset. Wobbrock et al. describe ten error types that are classifiable using their algorithm. As we did not need to account for non-recognition errors, we were able to use their algorithm to label each keystroke as one of the following error types:

- **Corrected No Errors** occur when a character is correctly entered in the input stream but subsequently deleted.

- **Corrected Omissions** occur when a character in the presented string is skipped in the input steam but later replaced.
- **Corrected Insertions** occur when a study participant deletes an extraneous character in the input stream that does not have a corresponding character in the presented string.
- **Corrected Substitutions** occur when a character in the input stream is substituted for its corresponding character in the presented string and then backspaced and corrected by the study participant.
- **Uncorrected Omissions** occur when a character in the presented string is skipped by the study participant.
- **Uncorrected Insertions** occur when a character is added to the input stream or transcribed string that does not exist in the presented string.
- **Uncorrected Substitutions** occur when a character is substituted for a corresponding character in the presented string.
- **Uncorrected No Errors** occur when a character in the presented string aligns with a corresponding character in the input stream or transcribed string with no issues.

Table 31 shows a breakdown of every character typed by the nine participants in the experimental stage of the evaluation in which the algorithm was turned on. These results are labeled as the ground truth against which we will measure the performance of FatThumbs.

9.3.1.2 Establishing FatThumbs ground truth

Next we proceeded to identify all off-by-one error types. These error types include off-by-one Roll-on errors, off-by-one Roll-off errors, off-by-one no errors, and instances

Error Type	**Quantity of Errors**
Corrected No Errors	107
Corrected Omissions	63
Corrected Insertions	34
Corrected Substitutions	77
Uncorrected Omissions	214
Uncorrected Insertions	245
Uncorrected Substitutions	451
Uncorrected No Errors	21,272

Table 31: The ground truth data labeled using Wobbrock et al's definition of the eight error types that were found in the experimental portion of the evaluation.

in which a tri-graph contains multiple off-by-one errors[2]. Given that the Wobbrockian ground truth data is determined solely from aligning characters between the presented string, input stream, and transcribed strings, at this point a limitation of our approach was uncovered. To determine our off-by-one ground truth data, we had to utilize not only character alignments but also keystroke timing. Doing so uncovered an new class of off-by-one error heretofore unclassified: the equal down-time error. An equal down-time error occurs when two keys are pressed at exactly the same time as measured by the Blackberry OS. In this case, it is impossible to properly align characters as neither of the two characters actually occurs prior the other. As such, an arbitrary decision has to be made about the order in which to write the keypresses out to the log file. Regardless of the order in which the characters appear, we logged and labeled each occurrence of an off-by-one equal downtime error along with every instance of the other off-by-one errors.

9.3.2 FatThumbs performance

Having labeled ground truth for each character with respect to off-by-one errors, I can now assess the performance of the FatThumbs algorithm. To do so, I track the

[2]As a reminder, we are not identifying off-by-one substitutions as our false positive correction rate is too high to be useful. We are also not including key repeats in the FatThumbs evaluation since key repeats occur so infequently when typing on modern Blackberry keyboards.

number of the times that FatThumbs performs an action. Each action we label a "trigger." The algorithm is triggered when it detects the presence of an off-by-one error. When triggered, FatThumbs ideally deletes the off-by-one error. We label this action a "correct detection." Unfortunately, the algorithm is not perfect and occasionally deletes a character that was correctly entered by the participant. Each time FatThumbs deletes a correct character, an "incorrect detection" occurs[3]. Occasionally an off-by-one error occurs and FatThumbs does nothing. Whenever this occurs we count a "missed detection." Table 32 presents FatThumbs performance for each error type.

In general, we are quite pleased with the performance of FatThumbs. FatThumbs successfully corrected 60.80% of the off-by-one errors committed by participants. It missed 37.54% of the errors and it false triggered 11.96% of the time. FatThumbs corrected 15.55% of the total errors committed by participants deleting 0.79% of the total keypresses entered during the experimental phase of the evaluation.

[3]Technically, in the case of an Equal Downtime error, "incorrect detections" should be termed "incorrect corrections" since Equal Downtime errors are not detected incorrectly. For simplicity, we report wrongly corrected Equal Downtime errors as "incorrect detections" to make comparing performance across error types easier.

Error Type	Quantity of Errors	Triggers	Correct Detections	Incorrect Detections	Missed Detections
OBO Roll–on	117	25	24	1	67
OBO Roll–off	173	134	102	32	46
OBO Multiple	11	0	0	0	0
OBO No Error	21,867	0	21,815	0	0
OBO Equal DT	0	60	57	3	0
Total OBO Errors	301	219	183	36	113

Table 32: FatThumbs performance. The table above shows ground truth quantities for each error type, the number of time that FatThumbs triggered, and the number of times a detection was correct, incorrect or missed. Of note, a correct detection occurs when FatThumbs correctly corrects an off-by-one error. An incorrect detection occurs when FatThumb incorrectly performs an action altering the input stream in the absence of an off-by-one error. A missed detection occurs when there is an off-by-one error in the ground truth but FatThumbs fails to perform any action.

9.3.3 Participant reaction

In addition to quantifying the impact of FatThumbs on users' typing performance, we were also interested in determining how participants felt about having their mistakes automatically corrected. The interview protocol was semistructured. I asked a consistent set of openended questions to each participant, prompting them to recall their experiences and to reflect on the challenges they encountered when typing with two thumbs on a mini-QWERTY keyboard. I also asked follow-up questions to pursue specific themes that arose in the context of the interview.

Towards the end of each interview, participants were told that I was actually evaluating an automatic error correction solution. Until that point in the evaluation, participants had not been informed about the true purpose for the evaluation. Their reaction to reading the debriefing information is captured as part of the interview.

In the interviews, I asked participants the following questions:

1. Can you discuss some of the challenges you encountered entering text into your mobile device?

2. At any point in the study did you notice your mobile device behaving erratically? Can you describe that behavior?

3. How do you feel about how you did in the study? Were you pleased with your performance?

4. In addition to exploring text entry performance, we were also investigating a novel spelling correction algorithm. Some of our participants evaluated this software, others did not. Which group were you in? How could you tell?

5. Did you realize that automatic error correction software was active? If so, did you find the automatic error correction helpful? Why or why not?

6. Do you have any comments, observations, or feedback for me or any other member of the research team?

One interesting finding from the interviews is how frequently participants described off-by-one errors on their own terms without previously being introduced to the idea. Six of the nine participants discussed off-by-one errors in one way or another. For example, when asked to discuss some of the challenges of entering text on a mini-QWERTY keyboard participant 1006 stated:

> Well, I guess the first thing that comes to mind, is just that I feel like I may have really big fingers, so I would occasionally try to press a button and I would accidentally press another button as well with that. So if I press C I would actually end up pressing C,V, or something like that.

The interviews also provided insight into participant error correction strategies. For example, when asked if the types of mistakes made by the participant changed over the course of the study, Participant 1007 responded:

> ...Most of my problems were pushing more than one key at once. So Id enter a couple of keys and then Id have to backspace if I wanted to do it correctly or if I wanted to just leave it. That was a decision I had to make.

When asked explicitly if they made a lot of corrections over the course of the study, Participant 1008 stated:

> The first half I did. The first four or five sessions. With that percentage going down as the sessions went on. Because I was like "I have to be perfect" and then I was like "No, thats ok" because my accuracy was still really good. I got above 94%, 93% and I was just focused on going faster.

Our goal with the interview was to determine if participants could notice and were bothered by the presence of our automatic error detection and correction solution. To that end, the most illuminating finding from the interviews is that, when informed that we were evaluating an automatic error correction system and asked whether or not they had been exposed to the system, all of the participants explicitly stated that they had not. Not one participant thought that they had been exposed to FatThumbs though each participant spend half of their final two sessions typing with FatThumbs enabled. This finding is all the more exciting as we were literally deleting characters that had been printed to the screen. Participants certainly had the opportunity to notice and be distracted by FatThumbs, but none of them reported noticing FatThumbs in action.

Full transcripts of the interviews can be found in Appendix D.

CHAPTER X

DISCUSSION AND FUTURE WORK

In this chapter, I summarize my findings, discuss the implications of my work and outline future directions.

10.1 Summary

In this dissertation I begin by describing three longitudinal studies designed to investigate typing on mini-QWERTY keyboards in various contexts of use. Next, I analyze the errors made in these evaluations and discover a common pattern in the types of mistakes that people make when typing on mini-QWERTY keyboards. The most common errors are off by one key to either the left or right of the intended key; I call these off-by-one errors. To improve typing on mini-QWERTY keyboards, I take a machine learning approach to create an algorithm to automatically detect and correct these errors. I refine the algorithm and reduce it to a set of simple rules that target feature of users' typing to automatically detect and correct off-by-one errors. Finally, I evaluate this set of rules on live typing by conducting a final longitudinal evaluation to assess the impact of automatic error correction on users' typing performance.

The first longitudinal evaluation (the Baseline evaluation, see Chapter 3 for details) collected baseline performance measures in an ideal environment. The Baseline evaluation was a between-subjects study that collected words-per-minute (WPM) and accuracy measures for fourteen participants (two groups of seven) typing for 20 twenty-minute sessions on two different keyboards. At the end of 400 minutes of typing, participants averaged approximately 60 WPM at approximately 95% accuracy. This study established that mini-QWERTY keyboards are the mobile text entry method that enables the fastest entry speeds with the least amount of training of any method on the market or in the literature.

	Baseline Study	**Blind Study**	**Mobile Study**
Date	Fall 2004	Spring 2005	Fall 2011
Participants	14	8	36
Expertise	Novice	Expert	Novice trained to expertise
Sessions	20	5	15 training, 15 in mobility conditions
Conditions	2	6	3
Phrases Typed	33,947	8,393	131,884
Keystrokes Typed	1,012,236	249,555	3,872,505

Table 33: The mini-QWERTY datasets for the Baseline, Blind, and Mobile Evaluations. In total, participants typed 174,224 phrases and 5,134,296 keypresses across all three evaluations.

In the second evaluation, I investigated the impact of limited visibility on expert mini-QWERTY keyboard text entry. Eight participants who had been trained to expertise input text for five 20-minute sessions in three different visibility conditions (the control condition termed the "normal" condition, the "hands blind" condition in which participants typed with their hands under a desk, and the "double blind" condition in which participants typed with both their hands under the desk and with obscured visual output). This evaluation established that when typing without being able to visually attend to the task of inputting text, participant performance decreases dramatically and does not recover to within a standard deviation of performance levels attained when participants input text with full visual attention.

My third longitudinal evaluation explores the impact of mobility on text entry performance. In this evaluation, 36 participants were trained to expertise and then transitioned to one of three different mobility conditions: walking, sitting, or standing. The order of conditions was balanced and participants proceeded to type for 100 minutes per condition. The mobility evaluation quantifies the WPM and accuracy rate decrease that occurs when participants enter text on mini-QWERTY keyboards while walking. Surprisingly, accuracy rates for expert typists decrease less than one percent (from approximately 95.3% to approximately 94.1%) while WPM rates only decreased four words-per-minute on average (from approximately 57 WPM to approximately 53 WPM). This evaluation establishes that there is, in fact, a significant decrease in WPM rates when users are mobile but shows that this decrease is much less than expected (< 10% impact on performance). Table 33 details the datasets for each of the three evaluations while Table 34 depicts the final WPM and accuracy rates for every condition in each evaluation.

Evaluation	**Condition**	**WPM (SD)**	**ACC (SD)**
Baseline Study	Dell	59.32 (SD=9.65)	91.29% (SD=5.32%)
	Targus	58.74 (SD=7.46)	94.68% (SD=2.09%)
Blind Study	**Normal**	57.89 (SD=4.80)	94.6% (SD=2.91%)
	Hands Blind	46.90 (SD=5.33)	85.2% (SD=7.43%)
	Double Blind	47.88 (SD=3.35)	84.8% (SD=5.64%)
Mobile Study	**Training**	48.21 (SD=13.11)	96.05% (SD=6.87%)
	Sitting	56.79 (SD=11.51)	95.36% (SD=6.15%)
	Standing	56.61 (SD=10.97)	95.25% (SD=6.54%)
	Walking	52.51 (SD=11.56)	94.91% (SD=6.59%)

Table 34: The average WPM and accuracy results for the final sessions of every condition in the Baseline, Blind Evaluations as well as the WPM and accuracy results averaged over the first 300 minutes of typing (the Training condition), and averaged over 100 minutes of typing in each experimental condition (Sitting, Standing, and Walking) in the Mobile Evaluation.

Upon completing the longitudinal evaluations, I investigated the types of errors committed by participants in various contexts. The evaluation uncovered that the most common error committed when typing on mini-QWERTY keyboards is what I term "off-by-one" errors. Off-by-one errors occur when an individual accidentally presses a character one key either to the left or the right of the intended character; in essence the error is off by one keystroke from the intended character. I compare errors made in the Baseline evaluation with those committed in the training phase of the Mobile evaluation and uncover how improvements in keyboard ergonomics essentially eliminate key repeat errors (though substitution errors occur at almost the exact same rate regardless of hardware). Finally, I investigate the impact of mobility on the types and frequencies of errors committed by participants in the

Mobile study. My analysis of mobile errors shows no difference in either the types or quantities of errors committed while mobile compared to those committed while seated or standing. In all cases there were statistically significantly more Substitution errors than the corresponding number of Insertion or Omissions errors committed. I show that as participants increase in experience, they correct far fewer mistakes than they do as novices. Finally, I demonstrate the existence of Semantic Errors (word level insertions) and show how they artificially negatively impact accuracy rates in text entry evaluations.

Having identified patterns in errors committed when typing on mini-QWERTY keyboards, I introduce Automatic Whiteout. Automatic Whiteout is a machine learning algorithm that leverages features of the users' typing to automatically detect and correct errors in mini-QWERTY typing. I trained and tested Automatic Whiteout over a variety of datasets from the Baseline and Blind evaluations in order to assess the performance of the algorithm across various levels of participant expertise, across different keyboards, and across different levels of visual attention to the keyboard. I demonstrate that Automatic Whiteout corrects approximately 25% of the off-by-one errors regardless of dataset. This result was surprisingly good given Automatic Whiteout does not use dictionaries for error detection and correction but instead relies only on tri-grams and bi-grams when making decisions.

Inspired by the success of Automatic Whiteout on previously collected data, I investigate the impact that automatic error detection and correction could have on live typing performance. To this end, I created a complete revision of my previous solution and introduced FatThumbs: a simple set of minimal rules designed to target off-by-one errors. Like Automatic Whiteout, FatThumbs leverages features of the users' typing to detect and correct off-by-one errors. Unlike Automatic Whiteout though, FatThumbs is built to run over live data, make decisions in less than 150 ms, and is able to make corrections after the first error is committed.

To evaluate FatThumbs, I conducted a fourth and final longitudinal evaluation of mini-QWERTY keyboard typing. I trained nine participants from novice to expertise over the course of 15 twenty-minute sessions. Participants then typed 90 phrases with FatThumbs enabled and 90 phrases with FatThumbs turned off. With FatThumbs enabled, participants typed significantly faster than when FatThumbs was inactive. Though participants created more errors when FatThumbs was on (perhaps due to their increased speed), FatThumbs was able to successfully correct enough errors (over 60% of the off-by-one errors) that the increase in errors did not impact accuracies.

I was interested not only in determining how FatThumbs impacted words-per-minute and accuracy rates but also in understanding if automatically correcting errors distracted or otherwise disturbed participants when inputting text on mini-QWERTY keyboards. To this end, I conducted the FatThumbs evaluation without alerting participants to the fact that I was correcting their errors. In interviews after the evaluation, all of the participants stated that they were unaware that errors were being corrected. When asked explicitly if they were in an experimental group that was exposed to my error correction solution, all of the participants declared that they were in a control group that was not exposed to automatic error correction. These results were particularly surprising given that FatThumbs was deleting characters printed to the screen. Ideally, FatThumbs will be implemented into the firmware of a mini-QWERTY keyboard. This approach would enable FatThumbs to simply intercept errorful characters after the keys are pressed but before they appear on the screen.

Over the course of this dissertation, I have conducted four longitudinal evaluations. Three of these evaluations explored text entry performance in a variety of contexts. I analyzed errors committed in these three evaluations and uncovered that off-by-one errors are the most common errors committed when typing on mini-QWERTY keyboards. I created and revised a solution for automatically detecting and correcting off-by-one errors in mini-QWERTY typing. When evaluating my solution in a final longitudinal study, I showed I could correct over 60% of the off-by-one errors and over 15% of the total errors in live typing on mini-QWERTY keyboards.

10.2 *Discussion*

In this section, I discuss my decision to conduct four longitudinal evaluations in which I recruit novices and train them to expertise. I investigate the merits of offline error correction and compare those results to the results from the FatThumbs live evaluation.

10.2.1 Training participants to expertise

The continued use of longitudinal evaluations in this dissertation sets it apart from other text entry work. Table 35 details of my training of novices participants to expertise.

	Baseline Study	**Blind Study**	**Mobile Study**	**FatThumbs**
Participants	17	8	36	9
Sessions	20	8	15	15
Minutes	400	200	300	300
Total Minutes of Training	21,900			

Table 35: Time spent training participants from novice to expertise. Three participants from the baseline study are excluded from analysis performed in Chapter 3. All training time in the Blind evaluation is excluded from analysis in Chapter 4. The "Total Minutes of Training" row displays the sum of Participants X Minutes for each evaluation.

Training participants for hundreds of minutes each is a time consuming, labor intensive activity which I ultimately believe is worth the effort. The contribution of the Baseline evaluation is the establishment of novice and expert WPM and ACC rates as well as participant learning curves. The value of this evaluation, fundamentally, is the training of participants. For the other four evaluations in this dissertation however, the obvious reason to invest the effort into training participants is to minimize any learning effects during the experimental phases of the evaluations. Training alleviates those learning effects that could be attributed to transcription typing with two thumbs on a mini-QEWRTY keyboard. In the Blind and Mobile evaluations, though I introduce new tasks to the participants in the experimental phase of the evaluation (Hands Blind, Fully Blind, Standing, and Walking tasks), participants continued typing in the control conditions without demonstrating learning. This result provides a strong baseline against which to compare results from the experimental conditions. To that end, in both the Blind and Mobile evaluations, when introducing the experimental conditions, participants still typed for 100 minutes per condition. Training participants in the new conditions helps minimize learning effects and provides cleaner results.

In the FatThumbs evaluation I introduced an error correction condition (FatThumbs On), and though one could argue that this task also new, I posit that there is no difference from the user experience perspective between the FatThumbs On and FatThumbs Off conditions. From the interviews I show that participants never noticed a new condition and their words-per-minute and accuracy results showed no learning in the FatThumbs Off condition.

Training novice participants to expertise also ensures that each participant has the same amount of experience typing with two thumbs. Had I chosen to recruit self described experts, participants' different levels of experience could have dramatically impacted results. For example, I removed a participant from the mobile study after discovering she used a iPhone as her primary mobile device. Preliminary analysis of her results showed that she was typing at 43 WPM after twenty minutes of typing. This rate obviously does not align with either my novice rates (approximately 30 wpm after 20 minutes of training) or my expert rates (60 wpm after 400 minutes of training). Perhaps recruiting self-described experts and training them for 100 minutes could be a compromise. This approach would reduce the training time dramatically while still ensuring that all participants had a similar degree of training. I am not convinced that this is an improvement on my current procedure but it would be interesting to investigate this approach.

10.2.2 Live Error Correction vs. Retrospective Results

One of the largest differences in the results between FatThumbs and Automatic Whiteout is the number of errors each solution is able to target. Automatic Whiteout is a retrospective analysis of previously collected data that targets Roll-on, Roll-off, Key Repeats, and Off-by-one Substitution errors. FatThumbs, on the other hand, operates on live data and targets Roll-on, Roll-off, and Equal Downtime errors. Section 7.3.1 describes how I sampled the previously collected datasets for the analysis. Of note, Automatic Whiteout only has the opportunity to correct the first error in a phrase whereas FatThumbs has the opportunity to correct all of the targeted errors as they are typed by participants. As such, while I have fewer off-by-one errors in the FatThumbs dataset, and I am detecting/correction fewer errors with the FatThumbs solution (not targeting Substitution errors which account for the greatest percentage of off-by-one errors), FatThumbs still successfully corrects approximately 60% of the off-by-ones and 15% of the total errors.

10.2.3 Improving FatThumbs

FatThumbs in its current incarnation performs well. However, there is still room to improve this solution. Obviously reducing the false positive rate for correcting off-by-one substitutions to the point where FatThumbs could be augmented to target off-by-one substitutions could dramatically improve the success of FatThumbs. Currently substitutions are not targeted because I am using tri-gram probabilities to determine which character to insert into a participant's input stream in case of a correction. I speculate that the reason that this approach has not proven successful so far is the mismatch between the QWERTY keyboard layout and tri-gram character frequencies in the English language. There are simply too many times when tri-gram frequencies predict the incorrect letter be inserted. One suggestion to mitigate this challenge that merits further investigation is to identify locations on the keyboard that have the highest level of ambiguity. For example, the **"e,r,t"** and **"y"** characters are often difficult to successfully differentiate when comparing tri-gram probabilities. Having identified areas of the keyboard where it is particularly challenging to correct off-by-one substitutions, I could choose to not make corrections when off-by-one substitutions are detected in these "hot spots" on the keyboard. I could correct off-by-one substitutions elsewhere on the keyboard and potentially reduce the number of false positives to the point where substitutions could once again be included in FatThumbs.

I believe that approaching substitution errors in this manner is potentially more beneficial than using a prefixing solution or relying on a dictionary to reduce the off-by-one substitution false positive rate. Using a dictionary-based solution has the potential to reduce false positives, but it makes mid-word correction nearly impossible. Mid-word correction is one of the key differentiators of FatThumbs. I was able to demonstrate that minor mid-word corrections can be made without distracting the user. Using a dictionary, on the other hand, introduces a delay between the time when the error is committed and when it is corrected. The longer the delay, the higher the cost of introducing a false positive (because the user will often have to delete more than one character to return to the location of the error), and greater the likelihood that the correction is noticed by the user. The simplicity of deleting the last character printed to the screen (or, if implemented in the firmware, of simply not displaying the errorful character) is one of the key benefits of FatThumbs. By correcting errors mid-word, FatThumbs has the potential to correct errors before the user notices that they have committed a mistake thus minimizing user distraction.

The ability to detect and correct errors without visually distracting the user (making a correction within milliseconds), enables faster rates of input and an improved user experience. Assuming that noticing error correction distracts or otherwise bothers users, in the future it would be interesting to compare the impact of mid-word error correction to word level error correction using dictionaries to quantify the impact that changing characters at different locations in a word has on the end user experience.

10.2.4 Decision Trees as a Visualization Tool

I arrived at the process used for transitioning from Automatic Whiteout to FatThumbs through a series of trials and errors. However, the method I employed, if executed with rigor, has the potential to generate viable results. At the high level, I took following steps in my approach to addressing errors in two-thumb typing:

1. Collect data from a user evaluation or multiple user evaluations
2. Identify characteristics of the data that merit investigation
3. Generate as many features as possible to investigate all aspects of those data
4. Select a machine learning approach to address the problem of interest (in my case, decision trees) [1]
5. Create an algorithmic solution that addresses the problem
6. Validate that solution by performing a retrospective analysis of the previously collected data
7. Tune thresholds and simplify the solution by continually testing the solution on the previously collected data

[1]It is worth noting that this procedure only works with machine learning techniques that descriptive (i.e. human readable). Decision trees work great but this method would not be successful with neural nets for example.

8. Implement the simplified solution and evaluate the impact of that solution in a new user evaluation

By following this approach, I was able to identify features of the users' typing that I would not have targeted had I employed only logical/obvious features in my solution. For example, on the surface, the duration of keypress (the time between the keydown and the keyup of the same keypress) might appear to be an obvious indicator of an error. The length of time that someone depresses a key is an obvious feature to target. However, duration is not one of the most discriminative features we investigated and I was able to use the above approach to identify features that are much more successful at error detection. When detecting and correcting Roll-off errors, FatThumbs checks to see if the current keyup time minus the previous keydown time is less than 170ms. I argue that this feature is not an intuitive feature to select and it is one that I would have never thought could be used to detect errors. However, it was a feature that was identified by using the above approach, and I used it successfully to identify and correct Roll-off errors in the FatThumbs evaluation. Decision trees help the HCI practitioner understand more about the problem without introducing biases of the practitioner. Specifically, by forcing the algorithm to make smaller and smaller decision trees that still performed well, I gained a better understanding of the mechanisms underlying how each error type was produced. Applied with rigor, I believe the methodology outlined above can be used to solve a wide variety of low-level human computer interaction challenges.

10.3 Future Work

Though I have performed a detailed investigation of typing on mini-QWERTY keyboards, there are still some under-explored areas which merit future investigation. For example, Wobbrock et al. demonstrated the potential of correlating instantaneous walking speed with text entry performance [17]. Inspired by this work, performing an analysis of errors that ties changes in instantaneous walking speed to participant error and correction behaviors is of interest.

Texting and driving is an underexplored area that is worthy of investigation. Evaluating the impact of driving on text entry performance is an open area of research, and it would be interesting to compare performance while driving to the walking performance I saw in the mobile evaluation. Can FatThumbs provide a benefit to drivers texting on mini-QWERTY keyboards?

Virtual mini-QWERTY keyboards are the most popular mobile keyboards in the market today. Replicating these studies (particularly the mobile study) using virtual mini-QWERTY keyboards would be interesting future work. Do the same types of errors occur on virtual mini-QWERTY keyboards? Is it possible to apply some of the automatic error detection and correction principles to virtual mini-QWERTY keyboards?

Perhaps most significantly, as I was not able to reduce the amount of false positives to the point where I was comfortable including off-by-one substitutions in FatThumbs, I would like to focus on ways of targeting off-by-one substitution errors more effectively. Off-by-one substitutions are the most common error made when typing on mini-QWERTY keyboards, and unfortunately my error correction solution does not address these errors with a high level of success.

I am interested in improving and extending my analysis of Semantic Errors. Some ways I could do this would be to address misspellings in semantic errors. Currently, I am only examining correctly spelled insertions; in the future I will allow for aligned search in the dictionary. One possible change to my approach could be to count the number of potential words that match the dictionary with less than a user defined minimum string distance. Taking this approach could yield a far greater number of Semantic Errors than I have currently been able to uncover.

Finally, I am interested in investigating the potential for automatic error detection to be used as a diagnostic tool. I am particularly interested in determining if individuals with dyslexia/dysgraphia commit error patterns that differ from the error patterns committed by those without these developmental reading disorders. If so, I think there is great potential to use a system similar to FatThumbs as an early stage detection solution. The potential to diagnose developmental reading disorders from errors in a user's two thumb text entry presents an exciting possibility.

CHAPTER XI

CONCLUSION

The work in this dissertation was conducted in an effort to support the following thesis statement:

FatThumbs, a method for automatically detecting and correcting typographical errors associated with pressing multiple keys at once in mini-QWERTY keyboard mobile text input, improves the text entry experience by reducing errors without distracting the user.

In this dissertation I present a series of experiments leading up to FatThumbs, a set of rules that automatically detects and corrects approximately 60% of the off-by-one errors (accounting for approximately 15% of the total errors) that occur in fixed-key mini-QWERTY keyboard text entry. FatThumbs has been shown to improve individuals typing performance by enabling faster typing rates without allowing a corresponding decrease in accuracy. It accomplishes this goal without distracting the user or perceptibly degrading the user experience.

In the process of developing FatThumbs, I conducted three longitudinal evaluations of text entry on mini-QWERTY keyboards. I investigated novice and expert performance as well as explored learning rates in the Baseline evaluation. I examined the impacts of limited visual feedback on typing performance and discovered that though individuals' can still type quickly and accurately when visual access to the keyboard or display is occluded, they are unable to perform as well as they can with full visual access to the keyboard and display. In essence, the Blind evaluation established that people are unable to touch type on a mini-QWERTY keyboard and expect results comparable to those they achieve when fully focused on the task of inputting text quickly and accurately. The third empirical investigation of mini-QWERTY keyboard text entry studied typing while in motion (the Mobile study). Though walking degrades performance, surprisingly, I observed less than a 10% decrease in WPM with no appreciable impact on accuracy. Neither the quantity nor the types of mistakes differ when comparing errors made while walking to those made while stationary.

Having collected over five million keypresses across these three evaluations, I then analyzed the errors in the studies and identified off-by-one errors as the most common error that occurs in mini-QWERTY typing. Off-by-one errors occur when an individual accidentally presses the key either one key to the left or the right of the intended key. These erroneous keypresses occur while trying to type the correct character as the user is pressing down or lifting off of the intended key with their thumb. Essentially, people have large thumbs and the keys on a mini-QWERTY keyboard are small and densely clustered resulting in the near simultaneous pressing of two keys at the same time.

In the final stage of my dissertation, I developed a solution to automatically detect and correct off-by-one errors that leverages features of the users' typing and I evaluated that solution on live typing. To accomplish this, I first performed a retrospective analysis of existing data from the Baseline, and Blind evaluations to address off-by-one insertion and substitution errors. This retrospective analysis, named Automatic Whiteout, was validated across different users, various levels of user expertise, different keyboard models, and various visibility conditions. On the whole, Automatic Whiteout was able to correct approximately 25% of total errors in the Baseline and Blind Datasets. Encouraged by this result, I optimized the solution for implementation on Blackberry hardware. This process resulted in a dramatic overhaul of my automatic error detection and correction solution. I reduced an algorithm that previously utilized a set of large decision trees designed to address each error class to a set of four simple rules, one for each error class. Each rule set is comprised typically of a timing threshold, the location of a keypress, and a language context rule. This revised solution is called FatThumbs. In a final longitudinal evaluation, FatThumbs was able to successfully address 60% of targeted errors resulting in an overall reduction of 15% of the total errors committed by participants.

APPENDIX A

FEATURES

Here is a list of all features used to train the decision trees used to visualize the mobile data.

- rollon False,True
- rolloff False,True
- repeat False,True
- obosubstitute False,True
- othererror False,True
- errortype 'nonerror', 'rollon', 'rolloff', 'repeat', 'obosubstitute', 'multiple', 'nonobo'
- isObo False,True
- userId NUMERIC
- sessionId NUMERIC
- pp_nom 'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z',' ','newline'
- p_nom 'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z',' ','newline'
- ascii_nom 'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z',' ','newline'

- next_nom 'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z',' ','newline'
- dt NUMERIC
- futdt NUMERIC
- prevdt NUMERIC
- futnowdiff NUMERIC
- nowprevdiff NUMERIC
- letterfreq NUMERIC
- prob1 NUMERIC
- futprob1 NUMERIC
- prob NUMERIC
- futprob NUMERIC
- bestneighbor_0 NUMERIC
- bestneighbor_1 NUMERIC
- bestneighbor_2 NUMERIC
- neighborfreq NUMERIC
- neighborfreqdiff NUMERIC
- neighborprob1 NUMERIC
- futneighborprob1 NUMERIC
- neighborprob1diff NUMERIC

- futneighborprob1diff NUMERIC
- neighborprob NUMERIC
- futneighborprob NUMERIC
- neighborprobdiff NUMERIC
- futneighborprobdiff NUMERIC
- dropprobdiff1 NUMERIC
- dropprobdiff1abs NUMERIC
- dropprobdiff1sign NUMERIC
- dropprobdiff NUMERIC
- dropprobdiffabs NUMERIC
- dropprobdiffsign NUMERIC
- dropprobgain1 NUMERIC
- futdropprobgain1 NUMERIC
- dropprobgain NUMERIC
- hdist NUMERIC
- vdist NUMERIC
- hdistabs NUMERIC
- vdistabs NUMERIC
- hpos NUMERIC
- vpos NUMERIC

- sameasprev_nom False,True
- ppisletter_nom False,True
- pisletter_nom False,True
- curisletter_nom False,True
- futisletter_nom False,True
- curfutadjacent_nom False,True
- prevcuradjacent_nom False,True
- dt_dd_0_p1 NUMERIC
- dt_du_0_p1 NUMERIC
- dt_uu_0_p1 NUMERIC
- dt_ud_0_p1 NUMERIC
- dt_dd_0_p2 NUMERIC
- dt_du_0_p2 NUMERIC
- dt_uu_0_p2 NUMERIC
- dt_ud_0_p2 NUMERIC
- dt_dd_1_0 NUMERIC
- dt_du_1_0 NUMERIC
- dt_uu_1_0 NUMERIC
- dt_ud_1_0 NUMERIC
- dt_dd_p1_p2 NUMERIC

- dt_du_p1_p2 NUMERIC
- dt_uu_p1_p2 NUMERIC
- dt_ud_p1_p2 NUMERIC
- dt_dd_p2_p3 NUMERIC
- dt_du_p2_p3 NUMERIC
- dt_uu_p2_p3 NUMERIC
- dt_ud_p2_p3 NUMERIC
- dur_1 NUMERIC
- dur_0 NUMERIC
- dur_p1 NUMERIC
- dur_p2 NUMERIC
- dur_avg_minus_dur NUMERIC
- dur_avg_minus_dur_nofut NUMERIC
- du_sub1 NUMERIC
- du_sub2 NUMERIC
- average_du_1 NUMERIC
- average_du_2 NUMERIC
- average_du_3 NUMERIC
- ud_sub1 NUMERIC
- ud_sub2 NUMERIC

- average_ud_1 NUMERIC
- average_ud_2 NUMERIC
- average_ud_3 NUMERIC
- uu_sub1 NUMERIC
- uu_sub2 NUMERIC
- average_uu_1 NUMERIC
- average_uu_2 NUMERIC
- average_uu_3 NUMERIC
- dd_sub1 NUMERIC
- dd_sub2 NUMERIC
- average_dd_1 NUMERIC
- average_dd_2 NUMERIC
- average_dd_3 NUMERIC

APPENDIX B

EXAMPLE OF A VERY SELECTIVE DECISION TREE

```
W-J48

J48 pruned tree
------------------

dt_uu_0_p1 ≤ 16
| prevcuradjacent_nom = False
| | futprob ≤ 0.003185:  nonobo (51.0/3.0)
| | futprob > 0.003185:  nonerror (79.0/13.0)
| prevcuradjacent_nom = True
| | curfutadjacent_nom = False:  rolloff (593.0/27.0)
| | curfutadjacent_nom = True:  multiple (32.0/5.0)
dt_uu_0_p1 > 16
| prob ≤ 0
| | prevcuradjacent_nom = False
| | | curfutadjacent_nom = False
| | | | sameasprev_nom = False
| | | | | futneighborprob ≤ 0.00578:  nonobo (569.0/17.0)
| | | | | futneighborprob > 0.00578
| | | | | | neighborprob ≤ 0.008696:  nonobo (178.0/20.0)
| | | | | | neighborprob > 0.008696
| | | | | | | average_dd_3 ≤ -160
| | | | | | | | futneighborprob1diff ≤ 0.000411:  nonobo (63.0/11.0)
| | | | | | | | futneighborprob1diff > 0.000411
| | | | | | | | | neighborprob ≤ 0.140351:  nonobo (54.0/25.0)
| | | | | | | | | neighborprob > 0.140351:  obosubstitute (38.0/6.0)
| | | | | | | average_dd_3 > -160
| | | | | | | | futneighborprobdiff ≤ 0.070866
| | | | | | | | | neighborprob ≤ 0.12:  nonobo (58.0/19.0)
| | | | | | | | | neighborprob > 0.12:  obosubstitute (75.0/16.0)
| | | | | | | | futneighborprobdiff > 0.070866:  obosubstitute (357.0/37.0)
| | | | sameasprev_nom = True:  repeat (87.0/22.0)
| | | curfutadjacent_nom = True
| | | | dropprobgain ≤ 0.009987:  nonobo (32.0/12.0)
| | | | dropprobgain > 0.009987:  rollon (263.0/30.0)
| | prevcuradjacent_nom = True:  rolloff (97.0/14.0)
| prob > 0
| | dt_ud_1_0 ≤ 144
| | | dropprobdiff1abs ≤ 0.001085:  multiple (30.0/16.0)
| | | dropprobdiff1abs > 0.001085
| | | | curfutadjacent_nom = False
| | | | | dropprobgain ≤ 0.009346:  nonerror (180.0/15.0)
| | | | | dropprobgain > 0.009346:  nonobo (85.0/27.0)
| | | | curfutadjacent_nom = True
| | | | | dropprobdiff ≤ 0.014085
| | | | | | dt_uu_1_0 ≤ 0:  nonerror (32.0/16.0)
| | | | | | dt_uu_1_0 > 0:  rollon (305.0/33.0)
| | | | | dropprobdiff > 0.014085
| | | | | | dropprobgain ≤ 0.005535:  nonerror (241.0/21.0)
| | | | | | dropprobgain > 0.005535
| | | | | | | dt_uu_1_0 ≤ 4
| | | | | | | | dur_avg_minus_dur ≤ -7:  nonerror (37.0/5.0)
| | | | | | | | dur_avg_minus_dur > -7
| | | | | | | | | neighborprob1diff ≤ -0.01676:  nonerror (33.0/12.0)
| | | | | | | | | neighborprob1diff > -0.01676:  rollon (30.0/10.0)
| | | | | | | dt_uu_1_0 > 4:  rollon (33.0/9.0)
| | dt_ud_1_0 > 144
| | | futprob ≤ 0
| | | | prevcuradjacent_nom = False
| | | | | sameasprev_nom = False
| | | | | | futneighborprob ≤ 0.00813
| | | | | | | curfutadjacent_nom = False
| | | | | | | | ud_sub1 ≤ -220:  nonobo (114.0/32.0)
| | | | | | | | ud_sub1 > -220
| | | | | | | | | prob ≤ 0.027027
| | | | | | | | | | nowprevdiff ≤ 32:  nonerror (42.0/16.0)
| | | | | | | | | | nowprevdiff > 32:  nonobo (43.0/12.0)
| | | | | | | | | prob > 0.027027
| | | | | | | | | | average_ud_2 ≤ -108
| | | | | | | | | | | futprob1 ≤ 0.004625:  nonerror (42.0/4.0)
| | | | | | | | | | | futprob1 > 0.004625
| | | | | | | | | | | | bestneighbor_2 ≤ 103:  nonobo (31.0/10.0)
| | | | | | | | | | | | bestneighbor_2 > 103:  nonerror (30.0/13.0)
| | | | | | | | | | average_ud_2 > -108:  nonerror (530.0/70.0)
| | | | | | | curfutadjacent_nom = True:  nonerror (55.0/24.0)
| | | | | | futneighborprob > 0.00813
| | | | | | | neighborprob ≤ 0.001684
| | | | | | | | du_sub1 ≤ -24:  nonobo (41.0/8.0)
| | | | | | | | du_sub1 > -24:  nonerror (66.0/11.0)
| | | | | | | neighborprob > 0.001684
| | | | | | | | dropprobdiff ≤ 0.209524
| | | | | | | | | futneighborprob ≤ 0.441176
| | | | | | | | | | average_du_3 ≤ -138
| | | | | | | | | | | futprob1 ≤ 0.03352:  obosubstitute (62.0/27.0)
| | | | | | | | | | | futprob1 > 0.03352:  nonobo (61.0/12.0)
| | | | | | | | | | average_du_3 > -138
| | | | | | | | | | | neighborprobdiff ≤ -0.015748:  nonerror (83.0/33.0)
| | | | | | | | | | | neighborprobdiff > -0.015748
| | | | | | | | | | | | p_nom = r:  obosubstitute (21.0/10.0)
| | | | | | | | | | | | p_nom = k:  obosubstitute (0.0)
| | | | | | | | | | | | p_nom = i:  obosubstitute (8.0/4.0)
| | | | | | | | | | | | p_nom = e:  obosubstitute (13.0/6.0)
| | | | | | | | | | | | p_nom = y:  obosubstitute (0.0)
| | | | | | | | | | | | p_nom =
| | | | | | | | | | | | | dur_avg_minus_dur ≤ 16:  nonerror (36.0/20.0)
| | | | | | | | | | | | | dur_avg_minus_dur > 16:  obosubstitute (32.0/13.0)
| | | | | | | | | | | | p_nom = w:  obosubstitute (8.0/3.0)
| | | | | | | | | | | | p_nom = a:  obosubstitute (32.0/6.0)
| | | | | | | | | | | | p_nom = t:  obosubstitute (6.0/2.0)
| | | | | | | | | | | | p_nom = c:  obosubstitute (8.0/1.0)
| | | | | | | | | | | | p_nom = h:  obosubstitute (2.0)
| | | | | | | | | | | | p_nom = d:  obosubstitute (17.0/1.0)
| | | | | | | | | | | | p_nom = n:  nonerror (9.0/4.0)
| | | | | | | | | | | | p_nom = m:  obosubstitute (3.0)
| | | | | | | | | | | | p_nom = o:  nonerror (8.0/5.0)
| | | | | | | | | | | | p_nom = v:  nonerror (1.0)
| | | | | | | | | | | | p_nom = s:  obosubstitute (10.0/6.0)
| | | | | | | | | | | | p_nom = z:  obosubstitute (0.0)
| | | | | | | | | | | | p_nom = u:  obosubstitute (5.0/1.0)
| | | | | | | | | | | | p_nom = g:  nonerror (5.0/1.0)
| | | | | | | | | | | | p_nom = l:  obosubstitute (6.0/2.0)
| | | | | | | | | | | | p_nom = p:  obosubstitute (4.0/1.0)
| | | | | | | | | | | | p_nom = f:  obosubstitute (8.0/3.0)
| | | | | | | | | | | | p_nom = b:  nonobo (3.0/1.0)
| | | | | | | | | | | | p_nom = j:  obosubstitute (0.0)
| | | | | | | | | | | | p_nom = x:  obosubstitute (0.0)
| | | | | | | | | | | | p_nom = q:  obosubstitute (0.0)
| | | | | | | | | futneighborprob > 0.441176:  obosubstitute (99.0/12.0)
| | | | | | | | dropprobdiff > 0.209524:  nonerror (47.0/13.0)
| | | | | sameasprev_nom = True
| | | | | | dt_du_0_p1 ≤ 100:  nonerror (42.0/11.0)
| | | | | | dt_du_0_p1 > 100:  repeat (34.0/17.0)
| | | | prevcuradjacent_nom = True
| | | | | prob ≤ 0.078652:  rolloff (40.0/17.0)
| | | | | prob > 0.078652:  nonerror (49.0/6.0)
| | | futprob > 0
| | | | sameasprev_nom = False
| | | | | prevcuradjacent_nom = False
| | | | | | futneighborprobdiff ≤ 0.019231
| | | | | | | futdt ≤ 68
| | | | | | | | dropprobgain ≤ -0.031088:  nonerror (94.0/3.0)
| | | | | | | | dropprobgain > -0.031088:  nonobo (31.0/13.0)
| | | | | | | futdt > 68
| | | | | | | | average_du_2 ≤ -148
| | | | | | | | | curisletter_nom = True:  nonerror (6977.0/250.0)
| | | | | | | | | curisletter_nom = False
| | | | | | | | | | prob ≤ 0.296
| | | | | | | | | | | dropprobdiff1abs ≤ 0.142045
| | | | | | | | | | | | dur_1 ≤ 108
| | | | | | | | | | | | | du_sub1 ≤ -164:  nonobo (33.0/11.0)
| | | | | | | | | | | | | du_sub1 > -164:  nonerror (33.0/10.0)
| | | | | | | | | | | | dur_1 > 108:  nonerror (36.0/5.0)
| | | | | | | | | | | dropprobdiff1abs > 0.142045:  nonerror (43.0/3.0)
| | | | | | | | | | prob > 0.296:  nonerror (505.0/23.0)
| | | | | | | | average_du_2 > -148:  nonerror (52001.0/661.0)
| | | | | | futneighborprobdiff > 0.019231
| | | | | | | neighborprobdiff ≤ -0.027027:  nonerror (6620.0/184.0)
| | | | | | | neighborprobdiff > -0.027027
| | | | | | | | neighborprob ≤ 0.428571
| | | | | | | | | dur_0 ≤ 52
| | | | | | | | | | futprob ≤ 0.092308
| | | | | | | | | | | hdistabs ≤ 0:  obosubstitute (31.0/15.0)
| | | | | | | | | | | hdistabs > 0:  nonerror (51.0/19.0)
| | | | | | | | | | futprob > 0.092308:  nonerror (92.0/17.0)
| | | | | | | | | dur_0 > 52:  nonerror (3125.0/322.0)
| | | | | | | | neighborprob > 0.428571:  obosubstitute (39.0/2.0)
| | | | | prevcuradjacent_nom = True:  nonerror (4125.0/67.0)
| | | | sameasprev_nom = True
| | | | | pisletter_nom = True:  nonerror (1874.0/47.0)
| | | | | pisletter_nom = False:  repeat (62.0/13.0)

Number of Leaves :  91

Size of the tree :  156
```

APPENDIX C

THE DOCUMENT SHARED WITH PARTICIPANTS DETAILING THE PURPOSE OF THE FATTHUMBS EVALUATION

Thank you so much for participating in our study. Here are some details about what we were studying and why:

Our study was designed to examine the impact of automatic error detection on mini-qwerty keyboard typing performance. In order to do this, we decided to train each of our participants until they were expert mini-qwerty typists. From previous studies, we have determined that it takes 300 minutes of typing on a mini-qwerty keyboard device to achieve expertise. As such, we had you type for fifteen 20-minute sessions using the phone. Once you had been trained to be an expert mini-qwerty typist, we put you (and everyone else) into the experimental phase of the study. For your last two typing sessions, instead of typing for 20 minutes, we had you type until you had completed 10 blocks of 10 phrases each (100 phrases). Half of those blocks randomly had our error correction algorithm turned on, half of them did not. So over the last two sessions, you typed 100 phrases with automatic error correction enabled and 100 phrases with it turned off. Everyone is going to go through the same procedure as you have. We did not mention or discuss correcting errors earlier because we wanted to see if our method was noticeable, and if it was, we wanted to learn how noticing the error correction impacted your performance.

Do you have any questions or comments for us about the study now that you know what we were specifically investigating?

Thanks again for your participation. We hugely appreciate it!

APPENDIX D

FULL INTERVIEW TRANSCRIPTS FROM THE FATTHUMBS EVALUATION

D.1 Participant 1001

Exit Interview (a semi-structured interview):

Can you discuss some of the challenges you encountered entering text into your mobile device?

> My nails got in the way a lot. Sometimes I would look at the keyboard and sometimes at the screen. Depending on which I was looking at it was hard to know if I made a mistake or to take more time to fix it. Sometimes my fingers would slide on the keys.
>
> In the beginning I was just looking at the keyboard. Towards the end I had a hard time deciding which one to look at. I typed faster if I looked at the screen but sometimes my eyes would wander towards the keyboard and look at it.

At any point in the study did you notice your mobile device behaving erratically? Can you describe that behavior?

> No, every once in a while it would stop and think but I was usually in the middle of typing so it never really effected anything.

How do you feel about how you did in the study? Were you pleased with your performance?

> I think did well. I enjoyed it. I think I would get faster but eventually I wouldn't have any motivation to get any faster. So I would just stop trying to type faster.

In addition to exploring text entry performance, we were also investigating a novel spelling correction algorithm. Some of our participants evaluated this software, others did not. Which group were you in? How could you tell?

> Spelling correction? I don't think it ever corrected my spelling. When I made a mistake I would have to go back and manually fix it.
>
> When I was staring at the keyboard I would look at what I was supposed to be typing and go down and type the wrong word or type an a instead of a v. That would be a common mistake I'd make. An then other times I would just type the letter next to the letter I was supposed to hit.

Did you realize that automatic error correction software was active? If so, did you find the automatic error correction helpful? Why or why not?

> Oh, that's why it seemed faster. (she said out loud while reading) That's cool.

Any questions for us?

> No, I think that is it.

D.2 1002

Exit Interview (a semi-structured interview):

1. Can you discuss some of the challenges you encountered entering text into your mobile device?

> The keys are really small. I think that was the biggest challenge just learning how to get used to hitting right in the middle of my thumb so

> it would just the key and nothing else around it. I think my fingernails kind of got in the way sometimes and they would hit the key above what I was trying to hit. But I think mainly the biggest challenge was the size of the keys.
>
> I don't think my process really changed. I just got faster and more accustomed to where the keys were. I was starting to be able to look more at the words I was typing and not at the letters. So my thumbs kind of adapted to where the keys were. I didn't have to look down to see where I needed to put my thumbs on each key. This is why I had the speed increase.

In the beginning were you looking more at the keyboard and at the end more at the screen?

> Yeah, it was probably half and half and that is why it felt so slow. And towards the end it was probably more closer to 75%-80% looking at the word and 20-25% looking at the letters.

Did you feel like you made a lot of mistakes?

> I think I was pretty accurate. I always try to leave my accuracy pretty high. I think every third or fourth block of ten phrases I would have to delete halfway through because I noticed I hit a "s" instead of a "a" I think I was fairly accurately.

2. At any point in the study did you notice your mobile device behaving erratically? Can you describe that behavior?

> Every now and then it seemed like there was a little hourglass that would pop up and then there would be delayed for just split second and then the

letters I was typing would catch up. But other than that, nothing[1].

3. How do you feel about how you did in the study? Were you pleased with your performance?

I think I did pretty well. I had pretty high number and accuracy and I could see steady increases in speed every time and I can only assume that that is pretty good.

What were things you felt like you did particularly well?

Well, I think the thing I did the most well was typing accurately because my accuracy was pretty high.

What was your accuracy all the way the through?

I think the lowest I ever got was 99.1 and I got a 100% one time so that was cool and exciting.

I think you were the only participant to do that so far.

Yes.. awesome, awesome.

4. In addition to exploring text entry performance, we were also investigating a novel spelling correction algorithm. Some of our participants evaluated this software, others did not. Which group were you in? How could you tell?

I was not in the one that had the spelling correction software.

How could you tell?

Because I spell things wrong.

[1]the hourglass appeared on the screen when Blacktwidor went into garbage collection mode. It introduced a very brief delay. Though minor, this delay was noticeable and several of the participants comment on it.

5. Did you realize that automatic error correction software was active? If so, did you find the automatic error correction helpful? Why or why not?

> Ohhh. ok. Interesting, ok, hmmm. I had no idea. I guess that's the point. Now that I think about it, I did notice that the sessions were shorter at the end.

D.3 1003

Exit Interview (a semi-structured interview):

1. Can you discuss some of the challenges you encountered entering text into your mobile device?

> Pressing multiple buttons at once. That was probably my biggest problem. Really early on, just cause I am touch typist I don't really think about where the keys are, I was not as familiar with the keyboard as I thought I might had been but that went away pretty quickly. I found after the first few sessions I could zone out and still be looking at the keyboard to see where my hands were but not to see if I was pressing "T". I didn't have to look at it.
>
> "B' on a regular keyboard kind of in the middle of the "G" and "T" but on this keyboard it was over to the side so a lot of time when I meant to press "N" I would press "B". I also had problems with the "Q" and "P" because they were really far on the edge and am not used to using the edge of the keyboard.
>
> Mostly in the beginning, it got a bit getting use to but near the end I wasn't too concerned about it.

2. At any point in the study did you notice your mobile device behaving erratically? Can you describe that behavior?

I think sometimes when I pressed space too early and I went to backspace to add a letter to a word it would delete the space plus the last letter that I typed. But that is like total speculation. But I think that might have happened, because I would press backspace twice to go back the space I had enter plus the last letter and then when I would enter the new letter then part of the word would be missing like it had deleted more then I thought I had.

3. How do you feel about how you did in the study? Were you pleased with your performance?

I'm not sure. You guys said I did well.

Were you satisfied with it?

Yeah.

Do you feel like you could have typed faster?

No, I tried pretty hard.

4. In addition to exploring text entry performance, we were also investigating a novel spelling correction algorithm. Some of our participants evaluated this software, others did not. Which group were you in? How could you tell?

I just texted entered things. So I don't think I evaluated the software.

So you were not in the experimental group, is what you are saying?

No, I don't think so.

5. Did you realize that automatic error correction software was active? If so, did you find the automatic error correction helpful? Why or why not?

I didn't notice. Some of them, I was like, there was no way that I got them 100% correct, but then it said I got 100% correct, and I was like whatever.

I had no idea.

D.4 1004

Exit Interview (a semi-structured interview):

1. Can you discuss some of the challenges you encountered entering text into your mobile device?

> First off, I would say that the initial challenge was just figuring out how to click the individual buttons, the size of the buttons, trying to... making sure you were accurate while still being quick. Learning the initial layout of the keyboard, certainly. But more the size of the buttons was the challenge for me.

How did the challenges changes over time?

> It got better. Every time I would try to ramp up the speed I would notice that the same challenges would come back a bit. So, I would step up the speed a little bit, and I would still keep missing them, then I would get better and I would be more accurate at that speed and then I would be able to step it up to the next speed. I noticed I was able to get each button individually pretty cleanly once I had practiced it a couple of times.
>
> Just by doing it. Almost slowing yourself down. I would try to maintain a reasonable speed that I thought I could do and work more on accuracy a little bit and then try to step it up to the next one. That would sometimes happen three times every 20 min sessions or towards the end I couldn't

get any higher. I tried to get more accurate. The end was kind of bad, to be fair.

I think I could do a little better. I think I could probably get it up two or three words per minutes more. But I think I pretty much was at the fastest that what my brain could keep up with my finger movement. I don't think I could do much faster than that. I think I could get more accurate. I don't think I could go that long at that speed. I got some in the 90s, some in the 70's some in the 60's but the average would be like 78 and I would say that I couldn't get higher than that average.

A lot of my mistakes were like dyslexic mistakes at first, and I got a little bit better. When I figured out "read it, memorize it and then spit it back out." Then some of those led to some dyslectic mistakes. At the end when I was trying to get really fast it was just my fingers wouldn't go to the right space. My finger just couldn't make it all the way to the edge and corner. I would hit the button next to it. Or I would miss spaces, I missed a lot of spaces. I would miss a lot of letters. And then after that the next letters were toast.

At first I deleted and went back to make sure my accuracy was good. And then I realized that it wasn't worth it. I stopped using delete. I gave up on mistakes. I basically told myself to just accept the fact that you missed it and go for the next letter as quickly as I could. Because if you just miss one letter or switch two letters in a row then your accuracy is still pretty good and it's still readable as well. I stopped worrying about it.

2. At any point in the study did you notice your mobile device behaving erratically? Can you describe that behavior?

I don't think it was the device. There was a couple of times that I thought

I clicked enter and I didn't. I wasn't sure if it was the phone not keeping up with me or whether it was me hitting two buttons simultaneously. I don't know why it did it. But it was certainly a couple of times when I thought I hit enter and it didn't happen. Other than that it was fine.

Talk about hitting buttons simultaneously. Did you do that often?

I did. I did every time I tried to speed it up that was like the initial learning curve I would hit two buttons. I obviously got better at it over time. It was more at first, I was able to correct that problem. The size of the buttons was actually very good even for my fingers which I guess are average size. It happened when I was trying to go too fast.

3. How do you feel about how you did in the study? Were you pleased with your performance?

I was pretty happy with it. I thought I did well. I thought that I wish I had tried to push myself earlier in it. I realized how quickly I could go. I really didn't have any clue. At first I was disappointed because I'm good on a desktop computer so I was expecting to be able to type pretty fast. But towards the end I was very happy with it. I like to get over 80. But we'll take it.

How fast do you think you type on a desktop computer?

I think I type about 110, 115.

4. In addition to exploring text entry performance, we were also investigating a novel spelling correction algorithm. Some of our participants evaluated this software, others did not. Which group were you in? How could you tell?

I think I was in the one that didn't have the error correction. It didn't change anything that I typed. So I typed it, it stayed, I guess.

Would you have liked to have been in that group.

> No, I would have hated it, probably. I don't know how the software works, but I think that would have slowed me down... for sure. And I mean, error correction, I have no idea what it's going to correct me to, necessarily. So, I liked not having it.. for sure.

5. Did you realize that automatic error correction software was active? If so, did you find the automatic error correction helpful? Why or why not?

> So I did have some error corrections in there? I didn't notice it. I didn't notice it at all. I guess I would be more focused on my typing. So I would look at the phase and then type it as quickly and accurately as I could. And then not look at the word per minute so I didn't even know if it was correcting or not. I would look at the accuracy number and it would still be about 100 so I'm like there's no error correction. I mean, I wasn't even thinking about it. I had no idea.
>
> There were definitely times I would hit two keys. Looking back on it,I actually do feel like there were some times, I guess you were saying it was just the last two times, that it was there. I felt like I was able to get one letter out a lot. I was like, ok that's odd, because I thought I hit two. But I didn't notice it. I would have never known it was doing a correction. I thought it was my correct type. I also noticed it a little bit before too, but I wasn't really paying attention what was showing up on the screen during the last half of the study. I was all about how quickly I could move on to the next one.

D.5 1005

Exit Interview (a semi-structured interview):

1. Can you discuss some of the challenges you encountered entering text into your mobile device?

> Typing, first of all, was one of them. Like, the buttons being... Figuring out a good way to position the fingers on the buttons, and not mashing down two buttons at one time and having to backspace. Very early on in the study, I was sometimes hitting "enter" instead of backspace so I would skip like the second half of the line and not being able to go back to it. It got easier later on, I don't know why. The other day I could remember in particular that the way I was going about it my fingers kept slipping on the buttons I guess because I was using my nails or something. But today and yesterday, I didn't notice at all.

Did you feel like you had the same challenges in the beginning as you did in the end?

> I think I got the hang of...I definitely got used to where the backspace and enter buttons were so I wasn't making the mistake of hitting enter prematurely anymore. And also, just typing on the keypad got a little bit easier.

Did you feel like you still made the same mistakes like pushing two keys at once?

> That still did happen.

Even as you got faster?

> Yeah, occasionally it would be just k's instead j's or something. It was still going on but less, not as frequent.

2. At any point in the study did you notice your mobile device behaving erratically? Can you describe that behavior?

> I didn't notice anything wrong with the phone itself. I didn't notice any technical problems.

3. How do you feel about how you did in the study? Were you pleased with your performance?

> I feel like I was pretty quick. There might be people faster than me though I don't feel like I was the best one.

What were some of the things you think you did really well?

> I feel like the accuracy stayed pretty high and was pretty consistent. I think that was a good thing that I did. I was usually in the same range of wpms too.
>
> I wanted to do better. There were times when I would hit 70 wpms but a few days ago I would get that once or twice. And I thought I would be reaching that more often but like today I still had some lines that were 40 (wpm). Because maybe I would make a mistake and have to backspace. Whenever you have to backspace something that just takes your time down a lot. I wasn't really getting in the 70's or high 60's.

Do you think you could type faster given more time or do you think you have leveled off?

> I would say I kind of leveled off. I think. I guess I was making progress up to this point so it's possible that I could still make some more progress.

4. In addition to exploring text entry performance, we were also investigating a novel spelling correction algorithm. Some of our participants evaluated this software, others did not. Which group were you in? How could you tell?

> You had an error correction software? Was I suppose to know this beforehand? What was the error correction thing do? Does it just know

> words so that if you hit a wrong key it would substitute in the letter it thinks... I think I was not under the error correction software. Because I feel pretty confident that I was hitting the right letters. Like the button mashing problems I was having where I would hit two buttons at the same time or maybe the wrong one it would still be there. I feel that an error correction software would avoid that.

5. Did you realize that automatic error correction software was active? If so, did you find the automatic error correction helpful? Why or why not?

> Interesting...I didn't notice it. No, I didn't.

Not a distraction?

> Not in the 100 phrases... yeah I didn't notice it. Yeah, what I was doing, I didn't really look back on my sentences very often. I would just kind of trust where my fingers were going and I would hit enter as soon as... I wasn't really checking so maybe I just didn't notice it was being implemented as I was typing.

When you were at the end of the study were you spending most of your time looking at the screen or looking at the keys, or where was your attention?

> Well, I would read the sentence and then yeah, my eyes would be on the keys. I would kind of just look at the center of the keyboard And I don't know, my eyes would maybe dart back and forth.

D.6 1006

Exit Interview (a semi-structured interview):

1. Can you discuss some of the challenges you encountered entering text into your mobile device?

> Well, I guess the first thing that comes to mind, is just that I feel like I may have really big fingers, so I would occasionally try to press a button and I would accidentally press another button as well with that. So if I press C I would actually end up pressing C,V, or something like that. I guess I also sometimes had a little bit of trouble figuring out which key was where, like I would forget where the X or the Z button was.

So talk a little more about the mistake when you would press two keys at once.

> I think that was a bit involuntary. It only happened some of the time. So I think it was just a matter of how my fingers were placed on the keys, so maybe it was more flat as opposed to straight down on the key. So that would usually happened, like I would pressing at a certain angle, so I would end up pressing the next key as well.

Did you challenges change throughout the study or were they pretty consistent all the way through?

> I would imagine they were pretty consistent. The study for me took pretty longer than most people decided to take it. I think I had a little trouble with it at the beginning but it definitely seemed more evident. It happened more often later in the study. Like today and the last time I was here, it definitely became evident that I became frustrated with myself.

Do you think that was because you were going faster as you went a long?

> That could be part of it. It definitely seemed like I got faster with the typing as the study went on.

2. At any point in the study did you notice your mobile device behaving erratically? Can you describe that behavior?

> There were times when the thing kind of paused like it was trying to registered what I typed in. I noticed that. I think that was pretty much it.

Did that happen consistently throughout or did that happen more in the beginning verses the end or vice versa?

> It was pretty consistent. I don't think there was a certain reason why it would it at times. It was just random moments.

3. How do you feel about how you did in the study? Were you pleased with your performance?

> Yeah, I thought I did pretty good. I would probably reconsider if I were to understand how well other people would do. Yeah I thought I did alright.

Do you feel like you could have typed a lot faster if the study was longer. Do you think you would have continued to improve?

> Maybe a little bit. It definitely seemed like I was starting to reach a bit of the plateau. I would usually get up to something around the 40 range. i would occasionally have the 50 whenever there was an easy ones, when you weren't typing really long words or something like that, the complicated ones.

4. In addition to exploring text entry performance, we were also investigating a novel spelling correction algorithm. Some of our participants evaluated this software, others did not. Which group were you in? How could you tell?

> I was the one without it. I just kept typing and it didn't really seem to correct anything.

5. Did you realize that automatic error correction software was active? If so, did you find the automatic error correction helpful? Why or why not?

> I guess I did kind of notice that a little bit, it was weird. I actually typed in something and then it backspace a little bit and I thought I actually pressed the backspace. Well you got me.

D.7 1007

Exit Interview (a semi-structured interview):

Can you discuss some of the challenges you encountered entering text into your mobile device?

> First off, I wasn't too familiar with the keyboard other than briefly using my dad's keyboard. The keyboard was kind of small and it was kind of hard figuring out the best way to use it. It just took time getting used to it more than anything else.
>
> Towards the end, I was able to do better because I had learned the keyboard and was able to move around faster on the keyboard. When I started out I was trying to figure out the best way to do it.
>
> I started out just using my left thumb for the left side of the keyboard and the right thumb for the right side but towards the end, if there were several letters in a row that were on the left side, then I'd use the other thumb and come over to just go faster instead of just using one thumb for each side.

Did you feel like that helped?

> A little bit, yeah. Because otherwise, you're just sitting there moving one thumb while the other thumb is not doing anything. Moving both means you have more usage of your thumbs.

Do you feel like you made many mistakes doing that?

> At the beginning, trying it out, I feel like I made a couple of mistakes. Just like pushing multiple keys at once. And then towards the end, I got better at it. More precise with the movement of my thumbs.

Do yo feel like you made the same mistakes all the way through the study or do you feel like those changed as you progressed?

> I feel like pretty much. Most of my problems were pushing more than one key at once. So I'd enter a couple of keys and then I'd have to backspace if I wanted to do it correctly or if I wanted to just leave it. That was a decision I had to make.

Do you feel like that decision changed over time?

> A little bit towards the end. I was just going more for speed than accuracy. If I did do a double, if I pushed two letters at once, I'd just let it go and just keep typing. Because I feel like it slows you down a lot when you go back. You're retyping everything twice and you have the backspaces on top of that. It really cuts down your ability to perform.

At any point in the study did you notice your mobile device behaving erratically? Can you describe that behavior?

> Only just briefly. When you're typing really fast and the little clock comes on the screen and spin for a second and then all the text would come out all at once instead of letter by letter. But that was only three or four times total.

Anything else?

> Not that I noticed, or distracted or bothered me or anything.

How do you feel about how you did in the study? Were you pleased with your performance?

> Yeah, absoultely. I saw a pretty good improvement going from 30 to 48. It is pretty exciting to see yourself improve and get better at it.

In addition to exploring text entry performance, we were also investigating a novel spelling correction algorithm. Some of our participants evaluated this software, others did not. Which group were you in? How could you tell?

> I feel like I didn't. You're saying it would correct spelling errors? I didn't see it correct anything so..I don't know. I didn't notice anything if I was.

Did you realize that automatic error correction software was active? If so, did you find the automatic error correction helpful? Why or why not?

> Now that you mention it, I vaguely remember that it might have switched once or twice. I didn't really think anything much of it because I was trying to go as fast as possible but I was like I think I might have changed once or twice. But I certainly didn't know it.

D.8 1008

Exit Interview (a semi-structured interview):

Can you discuss some of the challenges you encountered entering text into your mobile device?

> I've had a flip phone since high school and that had physical buttons but no but it was not a qwerty keyboard. So basically the only qwerty keyboard I have typed on that is that small is on a touchscreen. So it was weird actually pressing different buttons with my thumb. And I would accidentally press two buttons at once. Which on a touchscreen you don't

> have to worry about because it would opt to choose one. Whereas on the Blackberry sometimes I would make two letters press at the same time if that makes sense. And that was kind of hard to get use to where to put my finger so that I would press only one.

How did you deal with that?

> I bent my thumbs more and just kind of made them more pointy. I made them more perpendicular to the phone. That way I had more accuracy with what buttons I was pressing.

Did you find it easier or more difficult to type on a phone with the fixed-keys vs. your previous touchscreen experience.

> Well, actually I use to say that I hated Blackberries because my mom had one. But now if I had a Blackberry I would not mind because I'm really good on typing on it now. So I think I would say I prefer a physical one actually. But the only problems is that a Blackberry doesn't do everything an iPhone does and that's what I want. But yes, if iPhones had physical keyboards, I would choose the physical keyboard.

Did you have different challenges in the beginning of the study to when you finished?

> You mean different challenges from the first session to like now? Yeah, yeah, pretty much the button pressing, where I would press two buttons at once. As well as, this is unrelated to the typing part of it, but like reading the phrase that came up. Sometimes I would like misread it or I would rephrase it in my head the way I would have written it you know? Cause there's like just bias I guess for someone reads something and they

think of I would have said it differently. Or, like, I think you guys avoided using contractions because obviously you can't do an apostrophe so the one that said "a dog is the best friend of a man" in my head I typed it "dog is mans best friend" just with "m a n s" and I did that a lot, it didn't matter what session it was. But that, again, wasn't typing, that was just reading. But typing wise, yeah, it was just me typing more than one button at once. And then also I got the backspace and enter button confused a couple of times.

Can you describe things you think you did particularly well?

Well, this might be cheating but, I was good at remembering the phrases. So I would like see the first word and know what the rest of the phrase was and then just go from there. Instead of having to like type it, look, type it, and keep looking back and forth, I would just memorize it. So, I just got to know most of the phrases and I would say that helped me a lot.

So, did you spend most of your time look at the screen or at the keyboard?

At the keyboard. Definitely. I think the percentage would have gone down if I had answered that after the first session instead of this session but still, overall, I looked at the keyboard more.

Did you make a lot of corrections as you went through the study?

The first half I did. The first four or five sessions. With that percentage going down as the sessions went on. Because I was like "I have to be perfect" and then I was like "No, that's ok" because my accuracy was still really good. I got above 94%, 93% and I was just focused on going faster.

At any point in the study did you notice your mobile device behaving erratically? Can you describe that behavior?

How do you feel about how you did in the study? Were you pleased with your performance?

In addition to exploring text entry performance, we were also investigating a novel spelling correction algorithm. Some of our participants evaluated this software, others did not. Which group were you in? How could you tell?

> Is this like the iPhone autocorrect? No, I wasn't in that group. It would have been helpful if I had been. It could have saved me time correcting mistakes. Because then when all was said and done I wouldn't have wasted those one or two seconds. Which isn't that long in the grand scheme of things. But in there (looks at the study office) it is. Every second counts.

Talk to me a little bit more about the mistakes you made.

> Well, I could feel the mistakes in my fingers. You know? If I pressed two buttons at once, I could feel on the pads of my thumb that there were two buttons on my thumb at once that were being pressed. Occasionally, if I hit one button but it was just the wrong letter, somehow, I just knew I hit the "v" instead of the "b" and so I would check it just to make sure. And I was like "oh yea, I did, I messed up." And sometimes I thought I messed up and I didn't and then I wasted like half a second looking at that and I'm like "oh crap!" but I got to know where the keys were and I could do it without looking and stuff. So that was pretty cool.

Did you realize that automatic error correction software was active? If so, did you find the automatic error correction helpful? Why or why not?

Do you have any comments, observations, or feedback for me or any other member of the research team?

Thank you so much for your participation!

D.9 1009

Exit Interview (a semi-structured interview):

Can you discuss some of the challenges you encountered entering text into your mobile device?

> In the beginning, I've never owned a Blackerry and I've only used one like one time. So I didn't have any experience using a qwerty keyboard on a phone before. So that was, starting off, the hardest part was using my thumbs and trying to connect that to where I would put my fingers on a normal computer keyboard but that, like, went away real quick.

Did you have other challenges in the study?

> Once I started getting up past fifty words per minute it became very hard to type accurately. I was mashing the buttons so hard. I was typing faster than I could read the phrase and process it. I was typing faster than I could think. I was typing faster than I was doing those first two things so that slipped me a little bit.

Did you find that the challenges changed over the course of the study?

> Yeah, absolutely. In the beginning, it was more how to type on the keyboard. But towards the end, the challenge was trying to anticipate the letters as they were coming and trying to process it quickly enough to be able to type it fast enough.

At any point in the study did you notice your mobile device behaving erratically? Can you describe that behavior?

> Yes. On the first phase, whenever I would type it, I would almost always have a pause after hitting the enter button. Which didn't seem to effect the time, the wpm or anything but, after a while I started anticipating that. And then there were the time blips that would come up every now and then. Andmaybe once or twice I felt like it might have glitched when I was typing. But that could have been human error. I was typing so fast it was too hard to tell.

Were you pleased with your performance?

> Yeah. If you told me after the first two sessions that I would be able type over 50 wpm at 95% accuracy I would have told you that you were crazy. Cause, in fact I had some other friends who were doing the study and they told me they were typing like in the 50s and 60s and I was like "I can't even imagine what that must be like" and then I was doing that in the last two (sessions).

How did that feel to get to that point?

> Well, you know, I have major olympic fever right now. So it became sort of a sports, competition, quest for me. So yeah, it felt really good.

In addition to exploring text entry performance, we were also investigating a novel spelling correction algorithm. Some of our participants evaluated this software, others did not. Which group were you in? How could you tell?

> I did NOT have autocorrect. That would have been awesome. You know, that explains a lot about my friends. Since that was not told, it was weird comparing to my friends, in retrospect.

Weird in what way?

> I was comparing myself to them and they were saying they were typing 65wpm at 100% or 99% accuracy. And I was like, “I don’t think that’s physically possible.” You know, on the Blackberry you can only go so fast and there is just no way you can go that fast unless you really practice. And that’s basically what I was doing. So, that explains a lot.

So, you didn’t feel like it was awkward or weird to have the autocorrection turned on?

> Wait, what? I didn’t have it. What? Woh, I feel like an idiot now. I was sure that I didn’t have it.
>
> I remember when I was typing very quickly, I was just absolutely mashing buttons. It would automatically delete and have the right word. I thought I had just hit the delete button, I was going so fast. It never occurred to me that that is what it was.
>
> Autocorrection. Wow. I mean, wow. That’s really funny. I had no idea.
>
> I guess from the experiment standpoint, I’m not a conspiracy theorist for this experiment. I decided not to worry about the blips or anything. I just quickly put that out of my mind because I was just so focused on typing speed that I was just not remotely thinking about it.

REFERENCES

[1] Barnard, L., Yi, J. S., Jacko, J. A., and Sears, A., "Capturing the effects of context on human performance in mobile computing systems," *Personal Ubiquitous Comput.*, vol. 11, no. 2, pp. 81–96, 2007.

[2] Brewster, S., "Overcoming the lack of screen space on mobile computers," *Personal Ubiquitous Comput.*, vol. 6, no. 3, pp. 188–205, 2002.

[3] Brewster, S., Lumsden, J., Bell, M., Hall, M., and Tasker, S., "Multimodal 'eyes-free' interaction techniques for wearable devices," in *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, (New York, NY, USA), pp. 473–480, ACM, 2003.

[4] Bristow, H. W., Baber, C., Cross, J., Knight, J. F., and Woolley, S. I., "Defining and evaluating context for wearable computing," *International Journal of Human—Computer Studies*, vol. 60, no. 5-6, pp. 798–819, 2004.

[5] Butts, L. and Cockburn, A., "An evaluation of mobile phone text input methods," in *Proceedings of the Australsian User Interfaces Conference*, 2002.

[6] Chamberlain, A. and Kalawsky, R., "A comparative investigation into two pointing systems for use with wearable computers while mobile," in *ISWC '04: Proceedings of the Eighth International Symposium on Wearable Computers*, (Washington, DC, USA), pp. 110–117, IEEE Computer Society, 2004.

[7] Chipchase, J., Persson, P., Piippo, P., Aarras, M., and Yamamoto, T., "Mobile essentials: field study and concepting," in *DUX '05: Proceedings of the 2005 conference on Designing for User eXperience*, (New York, NY, USA), p. 57, AIGA: American Institute of Graphic Arts, 2005.

[8] Cisco, "Cisco visual networking index: Global mobile data traffic forecast update, 20112016," Februrary 2012. `http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.html`.

[9] Clarkson, E., Clawson, J., Lyons, K., and Starner, T., "An empirical study of typing rates on mini-qwerty keyboards," in *CHI '05: CHI '05 Extended abstracts on Human factors in computing systems*, (New York, NY, USA), pp. 1288–1291, ACM Press, 2005.

[10] Clarkson, E., Lyons, K., Clawson, J., and Starner, T., "Revisiting and validating a model of two-thumb text entry," in *In the Extended Abstracts on Human Factors in Computing Systems (CHI 2007)*, (New York, NY, USA), ACM Press, 2007.

[11] Clawson, J., Lyons, K., Rudnick, A., Robert A. Iannucci, J., and Starner, T., "Automatic whiteout++: correcting mini-qwerty typing errors using keypress timing," in *CHI '08*, (New York, NY, USA), pp. 573–582, ACM, 2008.

[12] Clawson, J., Lyons, K., Starner, T., and Clarkson, E., "The impacts of limited visual feedback on mobile text entry for the twiddler and mini-qwerty keyboards," in *Proceedings of the Ninth IEEE International Symposium on Wearable Computers*, pp. 170–177, 2005.

[13] Clawson, J., Rudnick, A., Lyons, K., and Starner, T., "Automatic whiteout: Discovery and correction of typographical errors in mobile text input," in *MobileHCI '07*, (New York, NY, USA), ACM Press, 2007.

[14] Dunlop, M. and Brewster, S., "The challenge of mobile devices for human computer interaction," *Personal Ubiquitous Comput.*, vol. 6, no. 4, pp. 235–236, 2002.

[15] Garner, S. R., "Weka: The waikato environment for knowledge analysis," in *Proceedings of the New Zealand Computer Science Research Students Conference*, pp. 57–64, 1995.

[16] Gentner, D., Grudin, J., Larochelle, S., Norman, D., and Rumelhart, D., *Cognitive aspects of skilled typewriting*, ch. A glossary of terms including a classification of typing errors., pp. 39–43. New York: Springer–Verlag, 1983.

[17] Goel, M., Findlater, L., and Wobbrock, J., "Walktype: using accelerometer data to accomodate situational impairments in mobile touch screen text entry," in *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, CHI '12, (New York, NY, USA), pp. 2687–2696, ACM, 2012.

[18] Gong, J. and Tarasewich, P., "A new error metric for text entry method evaluation," in *CHI '06*, (New York, NY, USA), pp. 471–474, ACM, 2006.

[19] Goodman, J., Venolia, G., Steury, K., and Parker, C., "Language modeling for soft keyboards," in *IUI '02: Proceedings of the 7th international conference on Intelligent user interfaces*, (New York, NY, USA), pp. 194–195, ACM, 2002.

[20] Grudin, J., *Cognitive aspects of skilled typewriting*, ch. Error Patterns in novice and skilled transcription typing, pp. 121–143. New York: Springer–Verlag, 1983.

[21] Hall, A. D., Cunningham, J. B., Roache, R. P., and Cox, J. W., "Factors affecting performance using touch-entry systems: tactual recognition fields and system accuracy," *Journal of Applied Psychology*, vol. 73, no. 4, pp. 711–720, 1988.

[22] Henderson, V., Grinter, R., and Starner, T., "Electronic communication by deaf teenagers," Tech. Rep. GIT-GVU-05-34, Georgia Institute of Technology, GVU Center, College of Computing, October 2005.

[23] James, C. L. and Reischel, K. M., "Text input for mobile devices: comparing model prediction to actual performance," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 365–371, ACM Press, 2001.

[24] Johnson, P., "Usability and mobility; interactions on the move," in *MobileHCI '98: Proceedings of the 1st Workshop on Human–Computer Interaction with Mobile Devices*, 1998.

[25] Kane, S. K., Wobbrock, J. O., and Smith, I. E., "Getting off the treadmill: evaluating walking user interfaces for mobile devices in public spaces," in *MobileHCI '08: Proceedings of the 10th international conference on Human computer interaction with mobile devices and services*, (New York, NY, USA), pp. 109–118, ACM, 2008.

[26] Kristoffersen, S. and Ljungberg, F., ""making place" to make it work: empirical explorations of hci for mobile cscw," in *GROUP '99: Proceedings of the international ACM SIGGROUP conference on Supporting group work*, (New York, NY, USA), pp. 276–285, ACM, 1999.

[27] Lin, M., Goldman, R., Price, K. J., Sears, A., and Jacko, J., "How do people tap when walking? an empirical investigation of nomadic data entry," *International Journal of Human-Computer Studies*, vol. 65, no. 9, pp. 759–769, September 2007.

[28] Lin, M., Price, K. J., Goldman, R., Sears, A., and Jacko, J. A., "Tapping on the move-fitts' law under mobile conditions.," in *Proceedings of Information Resources Management Association International Conference*, pp. 132–135, 2005.

[29] Lyons, K., Plaisted, D., and Starner, T., "Expert chording text entry on the twiddler one–handed keyboard," in *Proceedings of IEEE International Symposium on Wearable Computing*, 2004.

[30] Lyons, K., Starner, T., and Gane, B., "Experimental evaluations of the twiddler one-handed chording mobile keyboard.," *Human-Computer Interaction*, 2006.

[31] Lyons, K., Starner, T., Plaisted, D., Fusia, J., Lyons, A., Drew, A., and Looney, E., "Twiddler typing: One-handed chording text entry for mobile phones," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM Press, 2004.

[32] MacKay, B., Dearman, D., Inkpen, K., and Watters, C., "Walk 'n scroll: a comparison of software-based navigation techniques for different levels of mobility," in *Proceedings of the 7th international conference on Human computer interaction with mobile devices & services*, MobileHCI '05, (New York, NY, USA), pp. 183–190, ACM, 2005.

[33] MacKenzie, I. S., "A note on calculating text entry speed," June 2002. `http://www.yorku.ca/mack/RN-TextEntrySpeed.html`.

[34] MacKenzie, I. S., Kober, H., Smith, D., Jones, T., and Skepner, E., "LetterWise: prefix-based disambiguation for mobile text input," in *Proceedings of the 14th annual ACM symposium on User interface software and technology*, pp. 111–120, ACM Press, 2001.

[35] MacKenzie, I. S. and Soukoreff, R. W., "A character-level error analysis technique for evaluating text entry methods," in *NordiCHI '02*, (New York, NY, USA), pp. 243–246, ACM, 2002.

[36] MacKenzie, I. S. and Soukoreff, R. W., "A model of two–thumb text entry," in *Proceedings of Graphics Interface 2002*, Canadian Information Processing Society, 2002.

[37] MacKenzie, I. S. and Soukoreff, R. W., "Phrase sets for evaluating text entry techniques," in *CHI '03 extended abstracts*, pp. 754–755, ACM Press, 2003.

[38] MacKenzie, I. S. and Zhang, S. X., "The design and evaluation of a high-performance soft keyboard," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 25–31, ACM Press, 1999.

[39] Matias, E., MacKenzie, I. S., and Buxton, W., "Half-QWERTY: a one-handed keyboard facilitating skill transfer from QWERTY," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 88–94, ACM Press, 1993.

[40] Miller, G. A., "The magical number seven plus or minus two : Some limits on our capacity for processing information," *Psychological Review*, vol. 63, pp. 81–97, 1956.

[41] Mizobuchi, S., Chignell, M., and Newton, D., "Mobile text entry: relationship between walking speed and text input task difficulty," in *MobileHCI '05: Proceedings of the 7th international conference on Human computer interaction with mobile devices & services*, (New York, NY, USA), pp. 122–128, ACM, 2005.

[42] Newell, A. F. and Gregor, P., ""user sensitive inclusive design" in search of a new paradigm," in *Proceedings on the 2000 conference on Universal Usability*, CUU '00, (New York, NY, USA), pp. 39–44, ACM, 2000.

[43] Nicolau, H. and Jorge, J., "Touch typing using thumbs: understanding the effect of mobility and hand posture," in *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, CHI '12, (New York, NY, USA), pp. 2683–2686, ACM, 2012.

[44] Oulasvirta, A., Tamminen, S., Roto, V., and Kuorelahti, J., "Interaction in 4-second bursts: the fragmented nature of attentional resources in mobile hci," in *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, (New York, NY, USA), pp. 919–928, ACM Press, 2005.

[45] Pascoe, J., Ryan, N., and Morse, D., "Using while moving: Hci issues in fieldwork environments," *ACM Trans. Comput.-Hum. Interact.*, vol. 7, no. 3, pp. 417–437, 2000.

[46] Price, K. J., Lin, M., Feng, J., Goldman, R., Sears, A., and Jacko, J. A., "Motion does matter: an examination of speech-based text entry on the move," *Universal Access in the Information Society*, vol. 4, no. 3, pp. 246–257, 2006.

[47] Resarch, F., "Sms usage remains strong in the us: 6 billion sms messages are sent each day," June 2012. `http://blogs.forrester.com/michael_ogrady/12-06-19-sms_usage_remains_strong_in_the_us_6_billion_sms_messages_are_sent_each_day?cm_mmc=RSS-_-MS-_-1710-_-blog_Michael\%20O'Grady`.

[48] Roeber, H., Bacus, J., and Tomasi, C., "Typing in thin air: The Canesta projection keyboard – a new method of interaction with electronic devices," in *Extended Abstracts of CHI 2003*, ACM Press, 2003.

[49] Schildbach, B. and Rukzio, E., "Investigating selection and reading performance on a mobile phone while walking," in *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services*, MobileHCI '10, (New York, NY, USA), pp. 93–102, ACM, 2010.

[50] Sears, A., Lin, M., Jacko, J., and Xiao, Y., "When computers fade pervasive computing and situationally-induced impairments and disabilities," in *International Conference on Human Computer Interaction*, vol. 2, pp. 1298–1302, 2003.

[51] Silfverberg, M., "Using mobile keypads with limited visual feedback: Implications to handheld and wearable devices," in *Proceedings of Mobile HCI 2003*, pp. 76–90, 2003.

[52] Soukoreff, R. W. and MacKenzie, I. S., "Measuring errors in text entry tasks: an application of the levenshtein string distance statistic," in *CHI '01*, (New York, NY, USA), pp. 319–320, ACM, 2001.

[53] Soukoreff, R. W. and MacKenzie, I. S., "Metrics for text entry research: an evaluation of msd and kspc, and a new unified error metric," in *CHI '03*, (New York, NY, USA), pp. 113–120, ACM, 2003.

[54] Soukoreff, R. W. and MacKenzie, I. S., "Recent developments in text-entry error rate measurement," in *CHI '04*, (New York, NY, USA), pp. 1425–1428, ACM, 2004.

[55] Soukoreff, R. W. and MacKenzie, I. S., "Towards a standard for pointing device evaluation, perspectives on 27 years of fitts' law research in hci," *Int. J. Hum.-Comput. Stud.*, vol. 61, no. 6, pp. 751–789, 2004.

[56] Vadas, K., Patel, N., Lyons, K., Starner, T., and Jacko, J., "Reading on-the-go: a comparison of audio and hand-held displays," in *MobileHCI '06: Proceedings of the 8th conference on Human-computer interaction with mobile devices and services*, (New York, NY, USA), pp. 219–226, ACM Press, 2006.

[57] Wigdor, D. and Balakrishnan, R., "TiltText: Using tilt for text input to mobile phones," in *Proceedings of UIST 2003*, ACM Press, 2003.

[58] Wigdor, D. and Balakrishnan, R., "A comparison of consecutive and concurrent input text entry techniques for mobile phones," in *Proceedings of CHI 2004*, pp. 81–88, ACM Press, 2004.

[59] Wobbrock, J., *Text Entry Systems: Mobility, Accessibility, Universality*, ch. Measures of Text Entry Performance., pp. 47–74. San Francisco: Morgan Kaufmann, 2007.

[60] Wobbrock, J. O. and Myers, B. A., "Analyzing the input stream for character-level errors in unconstrained text entry evaluations," *ACM Transactions on Computer-Human Interaction*, vol. 13, no. 4, pp. 458–489, 2006.

[61] Yatani, K. and Truong, K. N., "An evaluation of stylus-based text entry methods on handheld devices in stationary and mobile settings," in *Proceedings of the 9th international conference on Human computer interaction with mobile devices and services*, MobileHCI '07, (New York, NY, USA), pp. 487–494, ACM, 2007.

[62] Zhai, S., Kristensson, P.-O., and Smith, B. A., "In search of effective text input interfaces for off the desktop computing," *Interacting with Computers*, vol. 17, no. 3, pp. 229 – 250, 2005.