Predicting the Cost of Error Correction in Character-Based Text Entry Technologies

Ahmed Sabbir Arif, Wolfgang Stuerzlinger
York University
Toronto, Ontario, Canada
{asarif, wolfgang}@cse.yorku.ca

ABSTRACT

Researchers have developed many models to predict and understand human performance in text entry. Most of the models are specific to a technology or fail to account for human factors and variations in system parameters, and the relationship between them. Moreover, the process of fixing errors and its effects on text entry performance has not been studied. Here, we first analyze real-life text entry error correction behaviors. We then use our findings to develop a new model to predict the cost of error correction for character-based text entry technologies. We validate our model against quantities derived from the literature, as well as with a user study. Our study shows that the predicted and observed costs of error correction correspond well. At the end, we discuss potential applications of our new model.

Author Keywords

User and cognitive model, performance metric, text entry, mobile phone, error correction.

ACM Classification Keywords

H.1.2 User/Machine Systems: Human factors.

General Terms

Human factors, measurement, performance.

INTRODUCTION

Text entry has become ubiquitous. The most popular text entry technologies are physical keyboards or on-screen keyboards operated by tapping with a finger or stylus [22]. There are also gesture- and speech-based technologies, and technologies that utilize handwriting. The later differ in nature as they are designed to process the input one word at a time, while most keyboards work at the level of one character at a time. The work reported here focuses on input technologies that are based on inputting a single character at a time, such as physical keyboards, on-screen keyboards, and gesture-based input technologies.

Error behavior in character-based text entry is not well understood. All existing models for the cost of error correction account, at best, for errors in an indirect way. They either fail to account for both human- and system-specific parameters or are not general enough to be used with different text input technologies.

We start this paper by analyzing human and system error correction behaviors in existing text entry experiments. Based on that, we present a model that accounts for both human- and system-specific parameters to measure and predict the cost of error correction. We then verify our model in several ways. First, we measure the cost of error correction for three popular text entry technologies, Qwerty, virtual keyboard, and 12-key mobile keypad, via data collected from previous work. Then, we present a user study that verifies if the predicted impact of system errors on Qwerty keyboard text entry matches real results. Finally, we generate some predictions and speculate on future extensions.

RELATED WORK

In the 1980's, Card *et al.* [5] presented the *GOMS* technique to predict user skilled performance time. They separated the human's cognitive architecture into four basic components: *goals*, *operators*, *methods* for achieving the goals, and *selection rules* for choosing among competing methods for goals. Despite the technique's power, it was never used on a large scale in the HCI community. The most likely reason is that the cost of first learning the technique and then constructing a correct model for a technology is too high to justify the benefits. Researchers have proposed different variations of *GOMS* to make modeling easier. For example, the Keystroke-Level Model (*KLM*) [4] eliminates all elements but the primitive operators. This makes *KLM* comparatively easier to learn and to construct models, but also makes it inadequate for multi-modal technologies. The Natural *GOMS* Language (*NGOMSL*) [13] is a high-level syntax for *GOMS* and based on cognitive complexity theory. Constructing *NGOMSL* models require performing a top-down, breadth-first expansion of the user's top-level goals into methods and further into primitive operators. Mastering the *NGOMSL* technique requires significant effort, as does the construction of a correct model. The Cognitive-Perceptual-Motor *GOMS* (*CPM-GOMS*) is based on a model human processor [7]. Unlike other variations of *GOMS*, *CPM-GOMS* is capable of modeling multitasking behaviors, because it does not enforce user interaction as a serial process. But *CPM-GOMS* also requires a thorough

CHI 2010, April 10–15, 2010, Atlanta, Georgia, USA.

understanding of *GOMS* and the human cognitive architecture.

ACT-R [1] is a cognitive architecture that aims to define both the basic, irreducible cognitive as well as perceptual operations that enable the human mind. As such, it looks like a programming language at first glance. Constructing an *ACT-R* model requires a detailed model of a cognitive task, which means significant amount of expertise, time, and effort. Moreover, the original form of *ACT-R* does not handle motor and perceptual systems correctly, although recent versions rectify some of the shortcomings.

To overcome the complexity of the model construction process, rapid modeling tools, such as *QGOMS* [3] and *CAT-HCI* [25], have been developed. The problem with these tools is that, once a model has been created, it is hard to change the model. Then, in case of upgrades or design changes, the developers have to either construct a new model or have to calculate the effect of that change by hand. Other tools, such as *CRITIQUE* [11], depend on research tools that are not commonly available. John *et al.* proposed a new system to overcome these problems by integrating *HTML* mock-ups with *ACT-R* [12]. This, however, limits the scope to web browsers.

There are several models that predict text entry speed or performance. But none of them account for the cost of error correction. The *KLM* model mentioned above can predict text entry performance, by counting keystrokes and other low-level operations, such as the mental preparation and the system's response time. A similar model was presented by Dunlop *et al.* [6] to forecast the performance of predictive text entry technologies, using three timing elements from *KLM*, though their timing elements were measured for full size keyboards. How *et al.* [10] improved that model by defining thirteen operators that map directly to operations on a mobile keypad for different text entry technologies. Later, Hollies *et al.* [9] presented another keystroke-level model to measure and predict mobile phone interactions. Their model considers even advanced interactions, such as using the embedded camera. Soukoreff *et al.* presented a theoretical model to predict upper and lower-bound entry speeds for using a stylus to tap on soft keyboards [23]. The model is based on the Hick-Hyman law for choice reaction time [21], Fitts' law for rapid aimed movements [21], and English linguistic tables for the relative frequencies of digrams. All of these models predict the performance of particular text entry technologies, without accounting for error correction methods and behavior. Several other metrics to characterize a techniques' performance exist. However, we do not discuss this here, as reviews of these metrics are available elsewhere [2, 26].

In 1997, Suhm developed an interactive multimodal error correction method for speech recognition [24]. His method can account for switches between different input modalities, such as continuous speech, oral spelling, hand-drawn gestures, choosing from a list of alternatives, cursive handwriting, or typing. To predict the performance of his technique he introduced a high-level model for the cost of error correction, as existing models cannot predict the performance of a multimodal technique. The model expresses the users' effort on error correction as a compound measure of the time required by the user to correct errors, the response time of the system, the accuracy of the automatic interpretation of corrected input, and the naturalness of the interaction. To overcome the model's technology dependency to some degree, Suhm separates human-specific parameters from system-specific ones. The model, however, does not contain important human-specific parameters, such as the visual verification time, human movement time, and the probability of committing an error. Moreover, the relationships between various parameters were not clearly explained.

Contributions

First, we present an analysis of real-life error correction behaviors and strategies. We used data from a previously reported experiment to analyze how error correction is carried out in text entry. We identified the most frequently used correction operation, the probability of making errors during the correction process, the effect of system-specific delays and errors, and so forth. We believe this analysis will help researchers and practitioners to understand the error correction process better.

We then use our findings to model error correction behavior. The initial purpose was to observe how different levels of accuracy, both human- and system-specific error levels, affect the error correction process and text entry itself. We verify our model through cross-comparisons with data from the literature. In addition, we describe different ways of calculating or deriving the parameter values for the model. Moreover, we performed an experiment to confirm the predictions. Some text entry technologies are not accurate, i.e. have significant system-specific error levels. We use our model to predict the effect of different level of system-specific errors and show that the experimental results correspond well to the predictions.

ERROR CORRECTION STRATEGIES

In text entry technologies, errors can be corrected with two different strategies: *character-level* or *word-level*. In character-level correction any erroneous character is corrected right away. In word-level correction, erroneous key presses are corrected after several other keystrokes have happened following the incorrect one. This kind of strategy is used when experienced users chunk their input, or when they do not verify their input right away. Almost all popular text entry technologies provide methods to correct errors that are committed with both strategies [8]. However, there is no data how frequent these two strategies occur in practice.

Data Collection

We obtained raw input logs from a recent text entry experiment [2]. There, the main purpose was to observe the effect of different error correction conditions on various text entry performance metrics, such as Words per Minute (*WPM*) and Keystroke per Character (*KSPC*), and to investigate the relationship between them.

In that experiment, 12 participants, aged from 22 to 45, average 27 years, participated. There were 3 sessions, 9 blocks in each session, with 20 phrases per block. The experiment used a Qwerty keyboard and the set of English phrases from MacKenzie *et al.* [17]. The participants were fluent English speakers and reasonably experienced touch typists (50 *WPM* or above).

That experiment analyzed three different error correction conditions: *none*, *recommended*, and *forced*. The *none* condition actively prevented participants from correcting errors, while the *forced* condition required uses to correct all errors. The *recommended* condition encouraged normal user behavior. That is, participants typed the presented phrase as fast and accurate as possible. Errors were corrected as users noticed them. Participants were informed that they could correct errors with their preferred method, including keyboard shortcuts, navigation keys, backspace, delete, or the mouse.

From the detailed event logs of the experiment, we extracted the data for the *recommended* condition. As the experiment used a counterbalanced design, we did not worry about asymmetric skill transfers. We then used the number of backspace characters in correction episodes to detect how quickly participants noticed and fixed errors. This was motivated by the fact that an analysis showed that 99% of the time participants corrected their errors with the backspace key. The remaining 1% was keyboard shortcuts, navigation and delete keys, or the mouse. We acknowledge that this high percentage of backspace usage may hold only for short English phrases and not for longer texts. However, other work shows that text *editing* is fundamentally different from text *entry* [20] and involves repeated problems solving and versatile planning. This makes editing unpredictable and hard to model. Thus, text entry correction in longer segments of text is virtually indistinguishable from general text editing. To remove this confound, we used data for short English phrases and all analysis was limited to error correction episodes involving backspace.

Overall, our analysis indicates that the proportion of error correction strategies is balanced. On average 50.29% (*SD* = 7.2) of all error correction efforts were character-level, and the remaining 49.71% (*SD* = 7.2) were word-level corrections. Figure 1 illustrates the different strategies of error correction by participant.

We further analyzed the word-level corrections to calculate more precisely when errors got noticed and corrected. Our results indicate that 96.10% of all times an erroneous character got noticed and fixed between the second and fifth character, counted from the erroneous one, and the rest got noticed within twelve characters. We also noticed that some errors were identified and corrected only when participants visually scanned the entered text *after* they were finished with the phrase. We did not consider these correction operations in our analysis, as this behavior occurred rarely, in 1.7% of all cases.

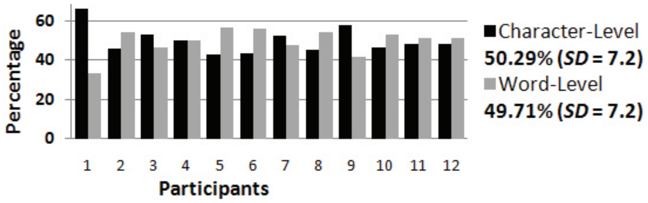

Figure 1. Character- and word-level error correction.

Sometimes errors happened during the error correction process. An example of such an incident would be the user first typing “b” instead of the desired “a”, and then erroneously typing “c” while attempting to fix the previous error. Our analysis shows that on average 6.68% (*SD* = 3.97) of the total errors were committed during the correction process. Of these, 86.11% were corrected in the second try, and the rest on the third iteration. We were not able to find an incident where more than three tries were required to fix an error.

THE COST OF ERROR CORRECTION

Error correction involves both human-specific elements, such as the time to verify a correction, as well as system-specific elements, such as the key sequence required to replace a wrong character.

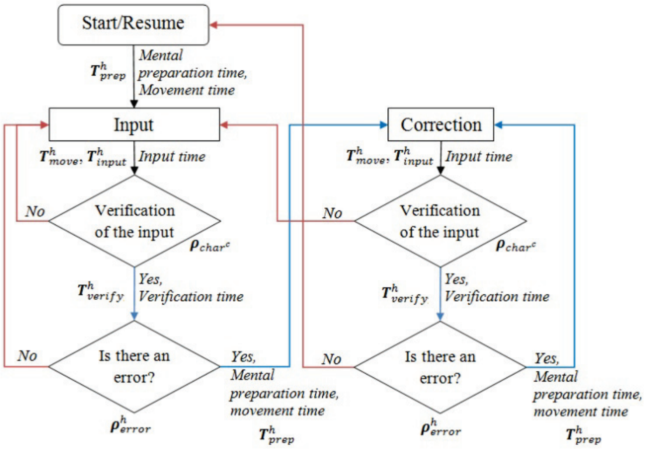

Figure 2. A flowchart representation of text entry error correction behavior.

Human Error Correction

It is necessary to have a better understanding of human error correction behavior to create a meaningful model. From our above analysis of error correction behaviors we found that the correction process follows a specific pattern.

Users usually immediately verify what they have typed and correct errors right away, i.e. character-level correction. However, users also chunk their input and verify the result only after typing a few characters or even the whole word. In the later scenario, denoted word-level correction, users navigate to the area where they have noticed an error, correct it, and then resume their work. As determined by our analysis of human error correction behavior, the predominant strategy is to use the backspace key for both character-level *and* word-level corrections! There seems to be no fundamental difference between the two strategies, except that rewriting multiple erased characters is an integral part of word-level correction, which scales linearly with the number of characters after which the error was noticed. Consequently, we see no reason to distinguish between character-level and word-level error correction behaviors in our model.

Error correction requires additional cognitive (planning) and motor (hand or finger movement) steps compared to error-free text entry behavior. Figure 2 illustrates the expected sequence of steps for normal text entry and error corrections in a flowchart. As illustrated, there are the following human-specific parameters for error correction:

- $\boldsymbol{T}^{h}_{prep}$, the *preparation time*, is the cognitive planning and decision time to start or resume a task.
- $\boldsymbol{T}^{h}_{move}$, the *movement time*, is the time required by the user to move their finger(s) to the intended key(s).
- $\boldsymbol{T}^{h}_{input}$, the *input time*, is the time necessary for the user to perform a single keystroke or similar operation, such as stylus tap, mouse click, or a gesture.
- $\boldsymbol{T}^{h}_{verify}$, the *verification time*, is the cognitive time required to verify that output matches input.
- $\boldsymbol{\rho}^{h}_{error}$, the *human-specific error probability*, is the probability of users making an error when performing a keystroke or similar operation.

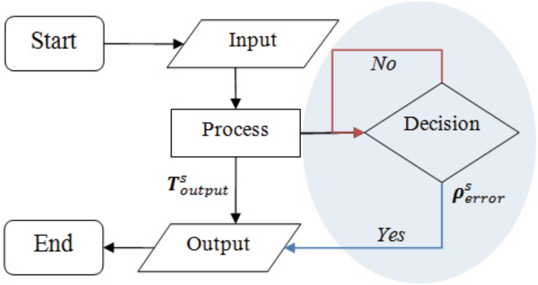

Figure 3. Input handling in text entry technologies.

System-Specific Parameters

Text entry technologies use both *open loop* and *closed loop* systems. In closed or feedback loop systems, inputs trigger the processes and the processes control the outputs. For example, while entering a character, a keystroke triggers the process that decides what character to display on the screen. On the other hand, in open loop systems, user inputs only trigger the processes that convert the input to the output, while the user can continue working. In other words, an open loop system does not directly monitor the output of the process that it is controlling. In some text entry technologies, such as handwriting or speech recognition, a specific command or operation may be processed in the background instead of immediately displaying the result. One potential reason is that e.g. many handwriting recognition methods cannot be performed fast enough. Figure 3 illustrates the input handling of the text entry technologies in a flowchart, where the shaded tasks are performed only in closed loop scenarios.

Depending on the technology and system, some recognition tasks may take significant time. That is why it is important to identify system specific parameters that may play a role in error correction procedures.

System-Specific Parameters

- ***s***, the *system*, is a specific text entry technology, defined by a combination of software and hardware. Examples are Qwerty, MultiTap, T9, etc.
- $\boldsymbol{T}^{s}_{output}$, the *system output time*, is the time necessary for the system ***s*** to process a keystroke or similar operations and output the result.
- $\boldsymbol{\rho}^{s}_{error}$, the *system-specific error probability*, is the probability of the system making an error when processing an input action. Some technologies are practically error-free or suffer only from very rare key malfunction. Other technologies, especially recognition technologies, have to distinguish between potentially fairly similar forms of input and exhibit significant error rates. For example, in handwriting recognition technologies, it is a common system error to misidentify a "u" as "v", and vice versa.

Compound Parameters

We define the following compound parameter based on the human and system-specific parameters defined above:

- $\boldsymbol{T}_{output}$, or *output time*, is the sum of the time to correct a character entered by the user and the time to process and display that input through the system.

$$\boldsymbol{T}_{output} = \boldsymbol{T}^{h}_{move} + \left(\boldsymbol{T}^{h}_{input} * \left(KSPC_f + 1\right)\right) + \boldsymbol{T}^{s}_{output} \qquad (1)$$

Here, $KSPC_f$ is the Keystrokes per Character (*KSPC*) metric calculated using a corpus's letter-frequencies. This metric is commonly used in text entry studies to measure the average number of keystrokes required to generate a character of text for a given text entry technology.

We scale $\boldsymbol{T}^{h}_{input}$ with $KSPC_f$ because there are technologies where it takes more or less than one keystroke to enter a character, e.g. MultiTap or T9. In *unambiguous* text entry technologies, such as Qwerty or Dvorak, it takes exactly a single keystroke to input a character. The "+ 1" term after $KSPC_f$ accounts for the fact that character-based text entry technologies have a dedicated delete or backspace key. This is justified, as 99% of all error corrections episodes used this backspace key to delete an erroneous character, see above. $KSPC_f$ can be calculated using the following equation [16]:

$$KSPC_f = \frac{\sum(\boldsymbol{K}_{char} * \boldsymbol{F}_{char})}{\sum(\boldsymbol{C}_{char} * \boldsymbol{F}_{char})} \qquad (2)$$

Here, $\boldsymbol{K}_{char}$ is the number of keystrokes required to enter a character, $\boldsymbol{F}_{char}$ is the frequency of the character in the corpus, and $\boldsymbol{C}_{char}$ is the length of the input, which is 1 for characters. $\boldsymbol{C}_{char}$ ensures that one can generalize this notion to input of more than a single character, e.g. words [16].

- $\boldsymbol{T}_{correct}$, the *correction time*, is the compound of the human and system time necessary to correct a single erroneous character in a single attempt. By adding the effort for the mental preparation necessary to fix an error to $\boldsymbol{T}_{output}$, we arrive at:

$$\boldsymbol{T}_{correct} = \boldsymbol{T}^{h}_{prep} + \boldsymbol{T}_{output} \tag{3}$$

We add the mental preparation time $\boldsymbol{T}^{h}_{prep}$ as error correction involves significant amount of cognitive planning and decision making, including the time to mentally “change tracks”.

The Probability of Errors

The whole probability for an error to happen can be expressed as:

- $\boldsymbol{\rho}_{error}$, the compound of the probability of either the system, or the user, or both making an error:

$$\boldsymbol{\rho}_{error} = (\boldsymbol{\rho}^{h}_{error} + \boldsymbol{\rho}^{s}_{error}) - (\boldsymbol{\rho}^{h}_{error} * \boldsymbol{\rho}^{s}_{error}) \tag{4}$$

Here, $\boldsymbol{\rho}^{h}_{error} * \boldsymbol{\rho}^{s}_{error}$ represents the probability of a simultaneous error by both the system and the user. We subtract this, as despite the fact that both parties have committed mistakes, such as the user entering a wrong character and the system misinterpreting that, everything results in only a single erroneous character.

The Probability of Noticing an Error $\boldsymbol{\rho}_{char}$

We discussed in a previous section that errors are not always noticed right after they were committed. In-depth analysis of the experimental logs indicates that the probability that an error will be identified after $\boldsymbol{c}$ characters is subject to exponential decay. This is illustrated in Figure 4, where we can also see that the data can be fit quite well with an exponential function ($R^2 = 0.97$). We speculate the out of line data point after the second character to be a behavioral pattern that does not follow the general trend. However, we do not have enough data to be able to make a definite statement on this.

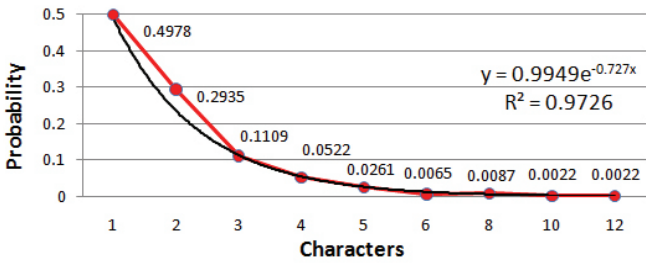

Figure 4. The probability of noticing an error after the *c*-th character $\rho_{char}{}^{c}$ is exponentially distributed.

Hence, we propose that the probability of noticing an error after $\boldsymbol{c}$ characters can be modeled accurately by an exponential distribution. More precisely, $\boldsymbol{c}$ is a nonnegative integer and $\boldsymbol{c} = 1$ means that the error was noticed right after committing it.

- $\boldsymbol{\rho}_{char}{}^{c}$, is the probability of noticing an error after $\boldsymbol{c}$ characters:

$$\boldsymbol{\rho}_{char}{}^{c} = a * e^{bc} \tag{5}$$

Where, $\boldsymbol{e}$ is Euler’s number ($\boldsymbol{e} \approx 2.718$), and a and b are constants that are determined by the curve fitting process.

A High-Level Model for the Cost of Error Correction

Based on the above analysis of error behavior logs, as well as the analysis of human error correction strategies and the relationship of human- and system-specific parameters we can present a new model for the cost of error correction:

- $\boldsymbol{T}_{fix_s}$, the average time it takes to fix errors using a text entry technology $\boldsymbol{s}$.

$$\boldsymbol{T}_{fix_s} = \sum_{i=1}^{\infty}\left(\boldsymbol{\rho}_{error}{}^{i} * \sum_{c=1}^{\infty}(\boldsymbol{\rho}_{char}{}^{c} * \boldsymbol{c} * \boldsymbol{T}_{correct})\right) \tag{6}$$

Here, $\boldsymbol{i}$ expresses the number of corrections. We multiply $\boldsymbol{c}$ with $\boldsymbol{T}_{correct}$ as the number of necessary correction operations increases with $\boldsymbol{c}$, the number of characters after an error was noticed. Notice that $\boldsymbol{T}_{fix_s}$ itself as well as the central term are both convergent series. Using the general formula for geometric sums we get:

$$\sum_{c=1}^{\infty}(\boldsymbol{\rho}_{char}{}^{c} * \boldsymbol{c} * \boldsymbol{T}_{correct}) = \frac{\boldsymbol{\rho}_{char}}{(1-\boldsymbol{\rho}_{char})^2} * \boldsymbol{T}_{correct} \tag{7}$$

Using Equation (7) and solving for $\boldsymbol{T}_{fix_s}$ we get:

$$\boldsymbol{T}_{fix_s} = \frac{\boldsymbol{\rho}_{error} * \boldsymbol{T}_{correct} * \boldsymbol{\rho}_{char}}{(1-\boldsymbol{\rho}_{error})(1-\boldsymbol{\rho}_{char})^2} \tag{8}$$

Equation (8) expresses the (extra) cost of error correction per character, in seconds or milliseconds. As values for $\boldsymbol{\rho}_{error}$ are likely smaller than 20% in any practically useful system and using a first order approximation, we can state that this means that that error correction effort is approximately directly proportional to the reliability of the user and the system, see also Figure 6.

The Cost of Error Correction vs. Error Correction Time

Note that $\boldsymbol{T}_{fix_s}$ does not predict the time it takes to correct n characters of errors, instead it predicts the *extra* time it will require on average per character to fix errors with a given text entry technology - regardless if a mistake was made on that character or not. For example, if we have x characters in a phrase then we can say on average it will take $\boldsymbol{T}_{fix_s}$ seconds more per character with text entry technology $\boldsymbol{s}$. $\boldsymbol{T}_{correct}$, on the other hand, predicts the time necessary to correct an erroneous character in a single attempt, see Equation (3). Therefore, designers can use this term as a measure for the error fixing time for a specific technology.

Limitations of the Model

Our model targets the cost of error correction in character-based technologies where texts are inputted one character at a time. As such, it cannot be applied directly to word-based technologies, such as, speech, gesture, or handwriting recognition, where texts are inputted word by word. This model will also not work without changes for multi-modal or predictive technologies and scanning keyboards. The reason is that there the process of entering text and correcting errors is different from character-based technologies. This model also does not account for environmental distractions, such as noise or motion. But we point out that practically all models for input tasks in HCI assume a distraction-free environment.

PARAMETER VALUES

Obtaining the necessary parameters for our new model typically requires controlled experiments. Some of our parameter values are largely independent of a specific technology. This is especially true for the human-specific ones. Other values can be collected from the existing literature on popular text entry technologies. However, some values need to be found experimentally, e.g. if new technologies are tested, or for existing ones that have not been well studied. To help in some of these situations, we present several alternatives to derive various parameters from commonly used text entry performance metrics.

Calculating $\boldsymbol{T}_{correct}$

The Words per Minute (*WPM*) metric is the most frequently used empirical measure of text entry performance. *WPM* measures how many words can be entered per minute. It is usually defined as:

$$WPM = \frac{|T|-1}{S} * 60 * \frac{1}{5} \qquad (9)$$

Here, *S* is time in seconds measured from the first key press to the last, including backspaces and other edit and modifier keys. The constant 60 is the number of seconds per minute, and the factor of 1/5 accounts for the length of a word [26], as it is common practice to regard a word as five characters including spaces, numbers, and other printable characters. Note that *S* is measured from the entry of the very first character to the last, which means that the entry of the first character is never timed, which is the motivation for the term "– 1" in the numerator of Equation (9).

We want to mention here again that a previous survey showed that text entry experiments are conducted with one of three error correction conditions [2]: *none* – where participants are not allowed to correct errors, *recommended* – where participants are asked to correct their errors as they identify them, and finally *forced* – where participants are forced to correct each error. But that study also showed that these correction conditions do not have a *significant* effect on *WPM*. Therefore, it is possible to approximate $\boldsymbol{T}_{output}$ from *WPM* using the following equation:

$$\boldsymbol{T}_{output} \approx \frac{60}{WPM*5} * \left(KSPC_f + 1\right) \qquad (10)$$

Here, $\frac{60}{WPM*5}$ is the time it takes to enter a character, see the derivation of Equation (9). We did not add the preparation time $\boldsymbol{T}^h_{prep}$, because it has already been accounted for in the *WPM* value. Moreover, *WPM* does not differentiate between the number of keystrokes made, the cognitive, or the motor time during text entry. Based on this approximation, we can estimate $\boldsymbol{T}_{correct}$ using Equation (3).

Calculating $\boldsymbol{\rho}_{error}$

It is standard practice to present error rates along with *WPM* when introducing or comparing text entry technologies. Error rates are usually calculated with one of four error metrics [2]: Error Rate (*ER*), Erroneous Keystrokes Error Rate (*EKS ER*), Total Error Rate (*Total ER*), and Minimum String Distance Error Rate (*MSD ER*). These metrics represent the combined errors committed by the human and the system. Hence, we can directly use these values as an approximation for the (compound) probability of committing an error.

The two error metrics *ER* and *MSD ER* are practically equivalent [2]. However, both do not consider the effort that was put into correcting errors. If users reliably corrected every erroneous character, these two metrics would still report the same value as if the text were entered error free from the start. *EKS ER* considers the cost of committing errors to some extent, but fails to show an accurate picture when some errors were not corrected. *Total ER* overcomes these shortcomings by computing the ratio between the total number of incorrect and corrected characters and the total effort to enter the text, providing more insight into the behaviors of the participants [2]. That is the reason *Total ER* yields a better approximation to $\boldsymbol{\rho}_{error}$ compared to the other alternatives.

Calculating $\boldsymbol{\rho}_{char}^{c}$

Previously we showed that the error recognition delay is well described by an exponential distribution. Hence, it is possible to calculate $\boldsymbol{\rho}_{char}^{c}$ using Equation (5), which is illustrated in Figure 4. There, one can see that almost 50% of all errors are noticed right after they are committed. We believe that this is a constant behavioral "constant", which does not vary across technologies. Hence, the data presented in Figure 4 together with the approximation $\boldsymbol{\rho}_{char}^{c}$ should be applicable to any text entry technology where text is entered one character at a time.

Parameter Values from Literature

We collected data from the literature to approximate the parameters necessary to compute $\boldsymbol{T}_{fix_s}$ for several popular character-based text entry technologies. The data and the respective sources are shown in Table 1.

The time it takes to perform a mental act depends on what cognitive processes are involved and is highly variable from situation to situation, or person to person. However, Kieras argued that it can be assumed that for routine thinking these pauses are fairly uniform in length [14]. Based on his argument we use the same preparation $\boldsymbol{T}^h_{prep}$ and verification $\boldsymbol{T}^h_{verify}$ time for unambiguous keyboards in Table 1. We also use the same value for the input time $\boldsymbol{T}^h_{input}$ for novices and experts for stylus-bases keypad technologies. This based on the observation [18] that there is probably only a small, perhaps negligible, difference between novices and experts in the motor act of tapping a key with a stylus. We did not add system-specific parameters to the table because system specific parameters are negligible in widely used text entry technologies. In particular, the reliability of keyboards is extremely high and the time to process and display a character is usually very low, at least compared to the human parameters.

How *et al.* performed an experiment to derive the "repeated keystroke time" and the "compound time of moving fingers and pressing a key" for 12-key keypads [10]. We subtracted the "repeated keystroke time" from the later to calculate the movement time $\boldsymbol{T}^h_{move}$. Finally, $KSPC_f$ is 1 for Qwerty, virtual (stylus-based) Qwerty, or similar keyboards, as they have dedicated keys for all characters.

s	**Qwerty or Dvorak**		**Virtual Stylus-Based Keyboards**		**12-Key MultiTap Keypad**	
Expertise	**Novice**	**Expert**	**Novice**	**Expert**	**Novice**	**Expert**
$\boldsymbol{T}^h_{prep}$	1.20[14]	0.60[14]	0.951[23]	0.60[14]	1.285[19]	0.60[14]
$\boldsymbol{T}^h_{move}$	0.40[4]	0.40[4]	0.40[4]	0.40[4]	0.96[11]	0.23[9]
$\boldsymbol{T}^h_{input}$	1.20[4]	0.12[4]	0.153[23]	0.153[23]	1.21[11]	0.39[9]
$\boldsymbol{T}^h_{verify}$	1.20[14]	0.60[14]	0.951[23]	0.60[14]	0.411[19]	.411[19]
$\boldsymbol{\rho}^h_{error}$	0.018[2]		0.0576[2]		0.091[2]	
$KSPC_f$	1		1		2.0342[16]	

Table 1. Human-specific parameter values for popular text entry technologies, collected from the literature. All timings are in seconds.

Prediction and Comparison

Based on the data in Table 1 we predicted the average time to fix an erroneous character $\boldsymbol{T}_{fix_s}$ for Qwerty, virtual, and 12-key mobile keypads. Not surprisingly, the 12-key MultiTap keypad requires the most time with $\boldsymbol{T}_{fix_s}$ of 0.7938 seconds per character, while the Qwerty keyboard has the lowest with $\boldsymbol{T}_{fix_s}$ of 0.096 seconds per character, see Figure 5.

As cross-validation, we also computed the average time to fix a character $\boldsymbol{T}_{fix_s}$ from measured *WPM* values [2], using Equation (10). The intent here was to observe if deriving $\boldsymbol{T}_{output}$ from *WPM* for $\boldsymbol{T}_{fix_s}$ gives a closer approximation to the original. The result is shown in Figure 5, where we can see that both calculations yield approximately the same result.

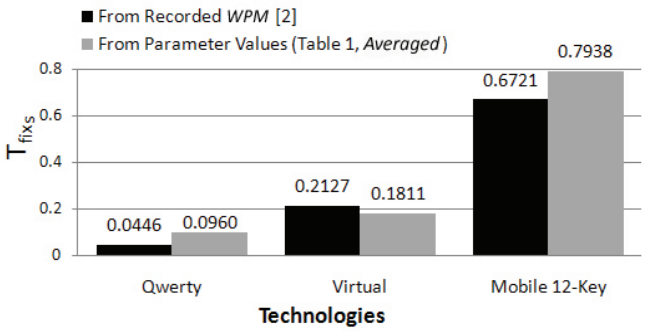

Figure 5. Comparison of different text entry technologies, calculated from collected data and *WPM*.

SYSTEM-SPECIFIC PREDICTIONS

We mentioned before that system-specific parameters are usually not significant in popular text entry technologies, as common text entry technologies process input and display the result in very small time frames. However, in some text entry technologies the system specific parameters may become an important factor.

Based on the data from Table 1, we gradually increased the system error rate $\boldsymbol{\rho}^s_{error}$ to analyze the effect of increasingly unreliable technologies on the time to fix an error $\boldsymbol{T}_{fix_s}$. Our analysis showed that $\boldsymbol{T}_{fix_s}$ increases approximately linearly as the probability of a system error increases. This is visualized in Figure 6.

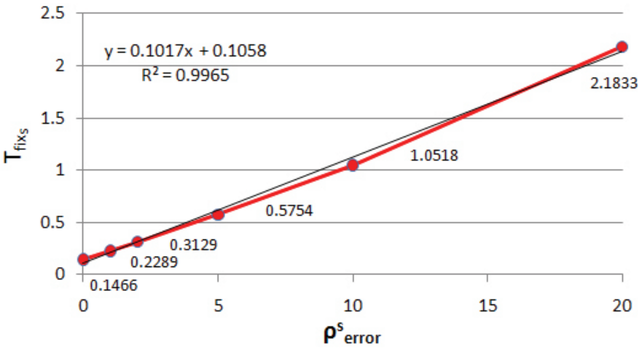

Figure 6. The increase in $\boldsymbol{T}_{fix_s}$ as $\boldsymbol{\rho}^s_{error}$ increases.

To verify this prediction, we conducted a user study to observe if this is true in a real-life. One can observe that the $\boldsymbol{T}_{fix_s}$ estimate for Qwerty keyboards with no system errors ($\boldsymbol{\rho}^s_{error} = 0$) is slightly higher in Figure 6 at 0.1466 seconds compared to the data in Figure 5, where it is 0.096 seconds. Figure 5 shows the cost of error correction by averaging $\boldsymbol{T}_{fix_s}$ values for both novice and expert users, while Figure 6 shows only $\boldsymbol{T}_{fix_s}$ values for novices. This assumes that routine thinking pauses for general, non-expert, users are very close to those of novices [14].

AN EXPERIMENT

The main purpose of this experiment is to observe the effect of system errors on the overall error fixing time.

Apertures

We used a Compaq *KB-0133* Qwerty keyboard and a 19" CRT monitor at 1280×960 for our study. A Java program logged all key presses with timestamps during text entry and calculated user performance directly. We used 15 point Tahoma font on the screen to present text.

Participants

Twelve participants from the university community, aged from 22 to 46 year, average 28 years, took part in the experiment. All of them had a minimum of 10 years Qwerty experience and three of them were touch typist. Three of our participants were female; all of them were right-hand mouse users. Participants were selected to be experienced typists and fluent English speakers to minimize learning effects during the experiment. Towards this, people with less than 8 years of typing experience were excluded from the experiment.

Procedure

During the experiment, participants entered short English phrases from MacKenzie *et al.*'s set [17]. This corpus was chosen because of its high correlation with the letter frequency in the English language. Moreover, these phrases are widely used in recent text entry studies.

Phrases were shown on screen to the participants in a dialog. They were asked to take the time to read and understand the phrases in advance, then to enter them as *fast* and *accurate* as possible, and to press the *Enter* key when they were done to see the next phrase. We also informed them that they could rest either between blocks, sessions, or before typing a phrase. Timing started from the entry of the first character and ended with the last, i.e., the character before the *Enter* keystroke.

Five system error rate $\boldsymbol{\rho}^{s}_{error}$ conditions were tested. The conditions had a predefined system error rate of 1, 2, 5, 10, and 20%. To imitate system error, the Qwerty keyboard input system was altered to output a pre-determined amount of erroneous characters, proportional to one of the five mentioned error rates. Although the amounts were pre-determined, the actual errors were generated randomly by replacing typed characters with surrounding ones on a Qwerty keyboard. For example, the character "h" was randomly replaced by one of the surrounding characters "y", "u", "j", "n", "b", or "g", and similarly for all other keys. The system error conditions were divided into five separate blocks that were randomly presented to the participants.

Participants were informed beforehand that the used keyboard is not 100% trustworthy and sometimes makes mistakes in interpreting the input. They were asked to work normally. That is, they should correct their errors as they notice them. They were also told that they could use any edit function, navigation key, or the mouse to correct errors.

We calculated the commonly used *WPM* metric to measure text entry speed. We also calculated *Total ER* to calculate error rates because this measure unifies the effects of accuracy during and after text entry [2], which gives us a more accurate picture of the error probability. The output time $\boldsymbol{T}_{output}$ was calculated by measuring the time interval between two consecutive keystrokes.

Design

We used a within-subjects design for the five system error conditions. There were three sessions. In each session participants were asked to complete fifteen blocks containing sixteen phrases, excluding two practice phrases. In each session the blocks were presented randomly to avoid asymmetric skill transfer. In summary, the design was: 12 participants × 3 sessions per participant × 5 blocks per session (i.e. the 5 system error conditions) × 16 phrases per block = 2880 phrases in total.

Results

The whole experiment lasted 45-75 minutes including the practice session, demonstration, and breaks. The highest and lowest typing speeds per block were 13 and 93 *WPM*. Similar to the data from the experiment [2] mentioned above, participants used backspace 99% of the time to correct their errors, even though they were able to use any edit operation, including keyboard shortcuts and the mouse.

Entry Speed and Error Rate

An ANOVA showed that there was a significant effect of different system error rates on both *WPM* ($F_{4,11} = 86.05$, $p < .0001$) and *Total ER* ($F_{4,11} = 787.61$, $p < .0001$).

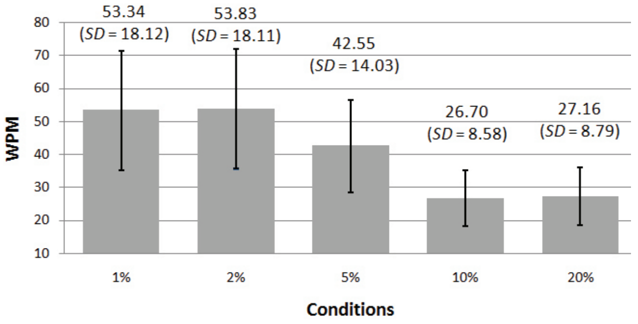

Figure 7. Average *WPM* for all system error rate $\boldsymbol{\rho}^{s}_{error}$ conditions.

A Tukey-Kramer multiple-comparison test showed that the 10 and 20% system error conditions had significantly lower *WPM* and higher *Total ER* compared to the 1, 2, and 5% conditions. As a reference, we recorded an average speed of 57.78 *WPM* (*SD* = 20) for text entry without system errors, i.e. 0% errors. This is higher than the performance levels for 1% and 2% errors, but not significantly so. Figure 7 and Figure 8 show the average *WPM* and *Total ER* measures, correspondingly, for all conditions.

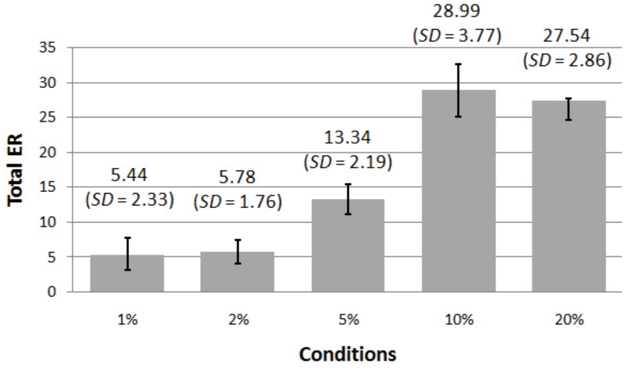

Figure 8. Average *Total ER* for all system error rate $\boldsymbol{\rho}^{s}_{error}$ conditions.

The output time (T_{output})

An ANOVA showed that there was a significant effect of different system error rates on $\boldsymbol{T}_{output}$ ($F_{4,11} = 15.54$, $p < .0001$). A Tukey-Kramer test showed that the 10 and 20% system error conditions had significantly higher $\boldsymbol{T}_{output}$ compared to 1, 2, and 5%. Figure 9 shows the average $\boldsymbol{T}_{output}$ for all conditions.

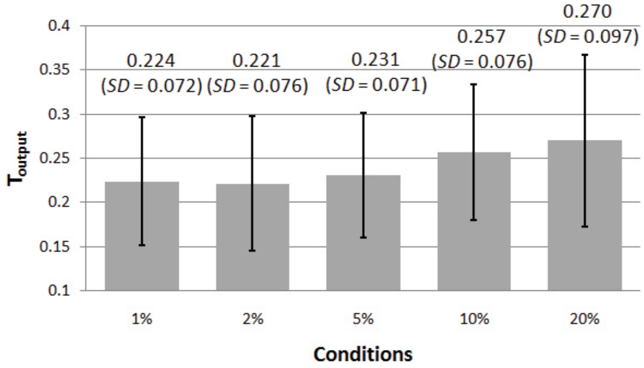

Figure 9. Average T_{output} for all ρ^{s}_{error} conditions.

System Error Analysis: Empirical Validation

The experimental data also corresponds well to our model's primary prediction: T_{fix_s} increases more or less linearly as the probability of a system error increases. Figure 10 visualizes this relationship. There, we can see that the experimental data fits a linear function reasonably well, with $R^2 = 0.9229$.

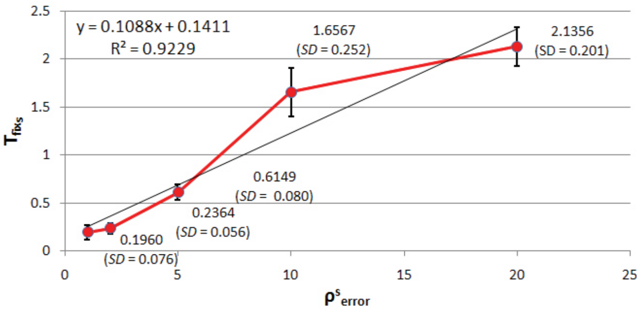

Figure 10. The increase in T_{fix_s} as ρ^{s}_{error} increases.

An ANOVA showed that there was a significant effect of different system error rates on T_{fix_s} ($F_{4,11} = 1108.42$, $p < .0001$).

DISCUSSION

The experimental results match the nature of predictions of the model: The data fits a linear approximation reasonably well. We currently do not know why fixing efforts were higher for the 10% value. One potential explanation is that participants may have treated both the 10% and 20% conditions in the same way – "just an unreliable system". We cannot directly compare data from our experiment with our initial data source [2], as the average *WPM* there is 75.84. Moreover, unlike our current experiment, that study screened participants for high typing speeds.

It is interesting to see that low error rates (1% and 2%) had no significant effect, even though the typing performance was somewhat lower. The most probable reason is that such low system error rates are indistinguishable from the average human error rates, e.g., 1.8% for expert typists [2]. We see this as an indication that keyboard failure rates of 1-2% are somewhat acceptable and have only a small effect on human performance, in the order of 7 to 8%. However, an error rate of 5%, i.e. 95% reliability, yields a noticeable drop in performance: 26%! A reliability of 80%, respectively 90%, approximately halves the input speed. This underlines how important reliability is for text entry technologies.

Another application of our model is to investigate what happens when various system parameters, such as the time to display a character, are changed. Properties of the input technology, such as the average number of keystrokes per character, also influence our model.

Generalization to Other Text Entry Technologies

We believe that our model and the predictions it generates are directly applicable to other physical keyboards, such as mini-Qwerty, Dvorak, and phone keypads. Assuming 100% reliable keys, only a derivation of the value of $T_{correct}$ for each distinct technology in necessary.

Another potential application area for our work is screen keyboards whose keys are too small to be hit reliably with a human finger, because the buttons are much smaller than the fingertip. There are currently many mobile phones that employ touch screens together with small screen sizes. Due to the lack of tactile feedback, such technologies are likely fundamentally different from mini-Qwerty keyboards. One could then model the ratio of the size of a fingertip relative to the displayed button size as a measure of keyboard reliability.

With this, we believe that it may be possible to predict the effect of varying button sizes in on-screen keyboards. Thus, it should be possible to predict some of the results of a recent evaluation [15] of touch screen keyboards, assuming $T_{correct}$ has been characterized.

CONCLUSION AND FUTURE WORK

In this article we investigated human error behavior in character-based text entry. We started by analyzing experimental logs from a different user study. We then created a new model for the cost of error correction. We verified our model against values derived from the literature as well as with a new experiment. We also discussed potential applications of our new model.

The current model is targeted at character-based text entry technologies. In the future, we would like to generalize the model to word-at-a-time input technologies, such as speech, gesture, or handwriting recognition. As the nature of error correction there is fairly similar – there is usually some form of "undo" operation, we are hopeful that our model can generalize to such technologies. Also, we plan to investigate the use of $T_{correct}$ as a new performance metric.

REFERENCES

1. Anderson, J. R., Bothell, D. and Byrne, M. D. An integrated theory of the mind. *American Psychological Association: Psychological Review 4*, 111 (2004), 1036-1060.
2. Arif, A. S. and Stuerzlinger, W. Analysis of text entry performance metrics. In *Proc. IEEE TIC-STH 2009*. IEEE New York (2009), 100-105.

3. Beard, D. V., Smith, D. K., and Denelsbeck, K. M. Quick and dirty GOMS: A case study of computed tomography interpretation. *Human-Computer Interaction 11*, 2 (1996), 157-180.
4. Card, S. K., Moran, T. P., and Newell, A. The keystroke-level model for user performance time with interactive systems. *Communications of the ACM* 23, 7 (1980), 396-410.
5. Card, S. K., Moran, T. P. and Newell, A. *The Psychology of Human-Computer Interaction.* Lawrence Erlbaum, Hillsdale, NJ, USA, 1983.
6. Dunlop, M. D. and Crossan, A. Predictive text entry methods for mobile phones. *Personal and Ubiquitous Computing 4*, 2-3 (2000), 134-143.
7. Gray, W. D., John, B. E. and Atwood, M. E. The precis of Project Ernestine or an overview of a validation of GOMS. In *Proc. CHI 1992*. ACM Press (1992), 307-312.
8. Grudin, J. T. Error patterns in skilled and novice transcription typing. In Cooper, W. E. ed. *Cognitive Aspects of Skilled Typewriting*, Springer-Verlag, New York, NY, USA, 1984, 121-143.
9. Holleis, P., Otto, F., Hussmann, H., and Schmidt, A. Keystroke-level model for advanced mobile phone interaction. In *Proc. CHI 2007*. ACM Press (2007), 1505-1514.
10. How, Y. and Kan, M.-Y. Optimizing predictive text entry for short message service on mobile phones. In *Proc. HCII 2005*, Lawrence Erlbaum (2005).
11. Hudson, S. E., John, B. E., Knudsen, K., and Byrne, M. D. A tool for creating predictive performance models from user interface demonstrations. In *Proc. UIST 1999*. ACM Press (1999), 93-102.
12. John, B. E., Prevas, K., Salvucci, D. D., and Koedinger, K. Predictive human performance modeling made easy. In *Proc. CHI 2004*. ACM Press (2004), 455-462.
13. Kieras, D. E. Towards a practical GOMS model methodology for user interface design. In Helander, M. ed. *Handbook of Human-Computer Interaction*, Elsevier, Amsterdam, Netherlands, 1988.
14. Kieras, D. E. *Using the keystroke-level model to estimate execution times.* Technical Report, University of Michigan, MI, USA (1993).
15. Lee, S. and Zhai, S. The performance of touch screen soft buttons. In *Proc. CHI 2009*, ACM Press (2009), 309-318.
16. MacKenzie, I. S. KSPC (keystrokes per character) as a characteristic of text entry techniques. In *Proc. HCI for Mobile Devices 2002*, Springer-Verlag (2002), 195-210.
17. MacKenzie, I. S. and Soukoreff, R. W. Phrase sets for evaluating text entry techniques. *Ext. Abstracts CHI 2003*, ACM Press (2003), 754-755.
18. MacKenzie, I. S. and Zhang, S. X. An empirical investigation of the novice experience with soft keyboards. *Behaviour & Information Technology*, 20 (2001), 411-418.
19. Pavlovych, A. and Stuerzlinger, W. Model for non-expert text entry speed on 12-button phone keypads. In *Proc. CHI 2004*, ACM Press (2004), 351-358.
20. Robertson, P. S. and Black, J. B. Structure and development of plans in computer text editing. *Human-Computer Interaction 2*, 3 (1986), 201-226.
21. Seow, S. C. Information theoretic models of HCI: A comparison of the Hick-Hyman Law and Fitts' Law, *Human-Computer Interaction 20*, 3 (2005), 315-352.
22. Silfverberg, M. Historical overview of consumer text entry technologies. In MacKenzie, I. S. and Tanaka-Ishii, K. eds. *Text Entry Systems: Mobility, Accessibility, Universality*. Morgan Kaufmann, San Francisco, CA, USA, 2007, 3-25.
23. Soukoreff, R. W. and MacKenzie, I. S. Theoretical upper and lower bounds on typing speed using a stylus and soft keyboard. *Behaviour & Information Technology*, 14 (1995), 370-379.
24. Suhm, B. Empirical evaluation of interactive multimodal error correction. In *Proc. IEEE ASRU 1997*, IEEE Signal Processing Society (1997), 583-590.
25. Williams, K. E. Automating the cognitive task modeling process: An extension to GOMS for HCI. In *Proc. HCII Poster Sessions*, Abridged Proceedings (1993), 182.
26. Wobbrock, J. O. Measures of text entry performance. In MacKenzie, I. S. and Tanaka-Ishii, K. eds. *Text Entry Systems: Mobility, Accessibility, Universality*. Morgan Kaufmann, San Francisco, CA, USA, 2007, 47-74.