

Recent Developments in Text-Entry Error Rate Measurement

R. William Soukoreff

Department of Computer Science
York University
Toronto, Ontario, Canada, M3J 1P3
will@acm.org

I. Scott MacKenzie

Department of Computer Science
York University
Toronto, Ontario, Canada, M3J 1P3
smackenzie@acm.org

ABSTRACT

Previously, we defined robust and easy-to-calculate error metrics for text entry research. Herein, we announce a software implementation of this error analysis technique. We build on previous work, by introducing two new metrics, and we extend error rate analyses to high key-stroke-per-character entry techniques, such as Multi-Tap.

ACM Classification Keywords

H.1.2. User/Machine Systems (Human Factors)

Keywords

Text entry, error rate, minimum string distance

INTRODUCTION

Three great peaks of text entry research are evident in the modern era. Yamada [14] provides an interesting and enlightening review of the first, in the late 1800s and early 1900s. This coincides with the invention of the typewriter, the development of touch-typing, and the eventual dominance of the Qwerty keyboard arrangement. MacKenzie [4] attributes the second peak to the arrival of the computer and the associated revolution in office automation throughout the world in the 1970s and 1980s. We are currently in the midst of the third peak in text entry research, which is precipitated by three trends. (1) The cellular telephone is immensely popular, and consequently more and more people around the world are sending SMS (phone-to-phone text) messages.¹ (2) Mobile computing is now in the mainstream, with sales of Personal Digital Assistants growing rapidly in recent years.² (3) Large corporations are seizing opportunities for growth in the mobile area, and are bringing more powerful portable devices to market, coupled with more convincing marketing. Further, market trends suggest that we are only at the beginning – the market penetration of these devices will rise for the next few years at least. With the increased use of mobile communications and computing devices comes a need for efficient text-entry techniques that work well on small devices, preferably supporting single-handed text entry, and ideally ‘eyes-free’ operation as well.

This huge demand for a portable text entry method caught the HCI research community unprepared. Not only is a solution to the mobile text entry problem lacking, but, until

recently, there were significant weaknesses in our research methodologies. This has not discouraged researchers from proposing new text entry methods. A recent review of mobile text entry methods [7] describes 23 novel text entry methods developed over the last decade, and 14 new soft-keyboard arrangements. Clearly, text entry is an active area of research. But, discerning which text entry method is superior requires controlled experiments with dependant variables that are comparable between studies. It was precisely this inter-comparability that was lacking.

Evaluations of new text-entry methods must attend to both speed and accuracy. Speed (in ‘words per minute’) is relatively easy to measure, but the same is not true for accuracy [10]. Typical text entry experiments generate large amounts of data that, in the absence of an automated means of analysis, are scanned for errors and tabulated by hand. Because computerising error analysis proved difficult and hand tabulation, tedious, many researchers either ignored errors, discarded trials with errors, or resorted to inferior experimental paradigms to avoid errors altogether. But, one cannot make meaningful observations about speed in the absence of accuracy. Further, error correction comprises a large part of the text entry process.³

Two recent publications address the difficulties in measuring accuracy in text entry studies [10, 11]. These provide a means to automate error analyses and define a family of useful performance metrics. The methodology is well regarded, however, through conversations with other researchers and from our own musings, some limitations are apparent. Specifically,

- Inspection of experiment data gives rise to new and potentially useful performance measures.
- Some clarification is needed on applying the methodology to certain (‘constructive’) text input methods.

¹ GSM World, reports that 30 billion SMS messages were transmitted in January, 2003. [3]

² eTForecasts, reports that worldwide sales of PDAs topped 15 million units in 2002, with a 28% annual growth rate. [2]

³ Card et al. [1] report that up to one fourth of an expert’s time can be spent correcting errors. MacKenzie et al. [7, 12] report that the backspace key is the second most common keystroke (following the space bar, but more common than the letter ‘e’) in typical desktop computer keyboard text entry.

- Some researchers have suggested that these error analysis techniques are prohibitively difficult to implement.

This paper addresses the three points above. We begin with an overview of the measurement of accuracy in text entry experiments. Two new error metrics are then presented, and the application of the error analysis methodology to constructive text entry methods is discussed. Finally, a software package is announced and released to the public that implements this error analysis methodology.

Accuracy in Text Entry

Typically during a text-entry experiment, short phrases are displayed one at a time to participants who enter each phrase using the text entry method under study. Thus, the data take the form of pairs of strings; there is the presented text string, and the transcribed text string (produced by the subject). These pairs of strings can be compared to determine how many errors the subjects committed. Pseudo-code for an algorithm that performs the comparison between the two strings is published, and is used to define the *MSD Error Rate*. [10]

A benefit in analysing errors in this manner is that researchers may employ a *correct as you go* procedure. This 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 correct mistakes as they go. However, the ‘correct as you go’ approach introduces a subtle wrinkle – it creates two classes of errors, those that the subject commits and then corrects, and those that go unnoticed and hence remain in the transcribed text. The latter (uncorrected errors) are measurable by the MSD error rate. The former (corrected errors) are not reflected in the final text, yet are an important aspect of the accuracy problem.

In later work, a new approach to measuring errors was described providing results consistent with the previous work, yet with better dependent measures for both corrected and uncorrected errors [11]. The new approach considers the keystroke input stream as well as the transcribed text. We 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, cursor movement).

A means to classify keystrokes is described and is readily automated [11]. Note that keystrokes in the *F* and *IF* classes do not appear in the transcribed text, because *F* keystrokes annihilate *IF* keystrokes. Thus, this analysis

requires access to the input stream (the exact sequence of keystrokes produced by the participant).

Several statistics are easily calculated once keystrokes are sorted into the above four groups, for example:

$$\text{Total Error Rate} = \frac{INF + IF}{C + INF + IF} \times 100\% \quad (1)$$

$$\text{Not Corrected Error Rate} = \frac{INF}{C + INF + IF} \times 100\% \quad (2)$$

$$\text{Corrected Error Rate} = \frac{IF}{C + INF + IF} \times 100\% \quad (3)$$

NEW STATISTICS

Both ourselves and other researchers [13] have observed that some participants engage in behaviour we term *pathologic error correction*. This is marked by extensive use of the backspace key instead of cursor keys when correcting errors. For example, consider the following input stream for entering “the quick brown”: (<’ represents backspace)

```
thw quick<<<<<<<<e quick brown
```

The participant committed an error but did not notice it until several keystrokes afterward. The participant then backspaced destructively to the error and re-entered the correct text. There are two points demonstrated in the example. Assuming the usual editing functions were available, the participant could have used ‘Control + Cursor Left’ to move to the beginning of ‘quick’ with only two keystrokes. And once the error was corrected, one tap of the ‘End’ key would return the cursor to the end of the line. In other words, the edit could have been made more efficiently (notwithstanding performance issues pertaining to human perception, but these are not addressed here). The second point is that we expect an increased error rate because the participant inflated the number of keystrokes required to complete the trial, and they might commit a further error when entering ‘e quick’ the second time.

Analysis of this example {*C* = 15, *F* = 7, *IF* = 7, *INF* = 0} reveals a *Corrected Error Rate* of 31.8%. On the other hand, if the subject noticed the error right away and corrected it immediately {*C* = 15, *F* = 1, *IF* = 1, *INF* = 0} the resulting *Corrected Error Rate* is as low as 6.3%. This difference arises because the *IF* keystrokes are defined as the keystrokes obliterated by corrections (*F* keystrokes). Yet in this example, *IF* contains many keystrokes that were correct (*IF* = {‘w’, space, ‘q’, ‘u’, ‘i’, ‘c’, ‘k’}, and so |*IF*| = 7) even though they were deleted by the participant.

Now, consider dividing the Incorrect Fixed keystrokes into two sets⁴: *IFc* contains the fixed keystrokes that were correct (*IFc* = {space, ‘q’, ‘u’, ‘i’, ‘c’, ‘k’}, and so

⁴ In practice, it is not too difficult to perform the separation of *IF* into *IFe* and *IFc* – for a detailed explanation see the source code for the ‘Analyse’ program described later in this paper.

$|IFc| = 6$), and IFe contains the fixed keystrokes that were in error ($IFe = \{ 'w' \}$, and so $|IFe| = 1$). This distinction allows us to break the Corrected Error Rate into two components,

$$\frac{\text{Corrected But Right}}{\text{Error Rate}} = \frac{IFc}{C + INF + IF} \times 100\% \quad (4)$$

$$\frac{\text{Corrected And Wrong}}{\text{Error Rate}} = \frac{IFe}{C + INF + IF} \times 100\% \quad (5)$$

where these terms sum to the usual *Corrected Error Rate* (as defined in [11]). So, the analysis for the example above still yields a *Corrected Error Rate* of 31.8%, but the *Corrected And Wrong Error Rate* would be 4.5%, and the *Corrected But Right Error Rate* would be 27.3%.

These new statistics provide a little more insight into the behaviour of subjects, and as such may be of interest to researchers.

CONSTRUCTIVE TEXT ENTRY METHODS

Constructive text entry methods are those that require several actions from the user to enter each letter. Examples include high *keystrokes per character (KSPC)* text entry methods like Multi-Tap on a cellular telephone, or predictive technologies such as T9 (by Tegic Communications, Inc. Seattle, WA; www.tegic.com). Multi-Tap and T9 are the two most popular input methods on mobile telephones, and as such many researchers seek to compare a new method to these two. However, it is not immediately clear how to apply this approach to error analysis with these input techniques. For example, when entering the four-letter word 'FILE' using Multi-Tap on a cellular telephone, the user must enter 11 keystrokes, and there is no one-to-one correspondence between keystrokes and characters in the input stream.

33344455533 ← keystrokes
F I L E ← entered letters

There are two ways to apply the error analyses. One can perform the prescribed calculations on the low-level raw keystrokes entered by the subject, or on the final resultant characters (high-level analysis). The correct approach is the high-level error analysis as the following examples demonstrate. (In the following examples, the presented text is the word 'FILE'.)

- a) *A missing keystroke:*
3344455533 ← keystrokes
D I L E ← entered letters

With a low-level keystroke error analysis, one keystroke is missing out of the 11, yielding an error rate of 9.1%, however 25% of the final letters are wrong. The low-level error analysis underestimates the error, but a high-level analysis (1 character incorrect out of 4) yields the correct result of 25%.

- b) *An extra keystroke:*

333244455533 ← keystrokes
F A I L E ← entered letters

Although 20% of the letters are incorrect, a low-level error analysis yields $(1 / 12 \times 100\% =) 8.3\%$. Because constructive text entry methods have a high keystrokes per character ratio, low-level analyses understate the effect of incorrect keystrokes. Yet each keystroke carries the ability to make a character incorrect, regardless of how high the keystrokes per character ratio is.

A further argument for performing the error analysis at the high-(character)-level, is that constructive methods frequently provide multiple ways to enter the same text, making error analysis difficult. For Multi-Tap, typically if one presses the '2'-key repeatedly, the selection wraps around to 'A' again.⁵ So, entering '2222' is equivalent to entering '2' – both mean 'A'. But this introduces extra keystrokes into the input stream that could dilute the effect of an error. For example:

- c) *An extra keystroke, and an inefficiently entered 'F':*
3333333244455533 ← keystrokes
F A I L E ← entered letters

Here the user used a circuitous method of entering 'F'. In this case, 1 incorrect keystroke out of 16 yields a keystroke-level error rate of 6.3%. Yet this same error committed in (b) above yielded a keystroke-level error rate of 8.3%. The correct but inefficient means of entering the 'F' should not adversely affect the final error rate. The character-level error rate is correct for this example at 20%, as it was in (b) above.

The arguments presented above are not meant to dissuade researchers from performing their own *ad hoc* keystroke-level analyses. Keystrokes per character, calculated on a character-by-character basis, and as an average, is useful for comparing text entry methods [5, 7 section 3.4]. Additionally a character-by-character analysis using a confusion matrix [6] may reveal where further *ad hoc* analyses should be focused. For the particular case of Multi-Tap, Pavlovych and Stuerzlinger [9] perform some illuminating analyses in their evaluation of a new text entry method similar to, but more efficient than, Multi-Tap.

ANNOUNCING A NEW SOFTWARE TOOL

We have heard from some researchers that the difficulty of implementing this error rate analysis methodology has discouraged some from using it. To make this methodology easier to use, and with the goal of improving the consistency of reported error rates, we have made our text entry experiment and analysis software available to the research community. The software has been released under the GNU Public Licence, and is available from

⁵ For example, a '2' keypress enters an 'A', '22' – 'B', and '222' – 'C', but '2222' produces the numeral '2', and '22222' wraps around to 'A' again.

<http://dynamicnetservices.com/~will/academic/textinput>. This software was written in Java, and has been verified to execute correctly in Windows and Linux environments (it should work on other Java platforms as well). The two primary Java classes are `Experiment` and `Analyse`. The `Experiment` class implements a fully-featured text entry experiment. The experiment software is flexible, yet easy to configure. The presented-text phrases (supplied by the experimenter, or from [8]) are presented to the subject one at a time, in a random order. As the participant enters phrases, keystrokes are time-stamped and logged for subsequent analysis. The presented text is displayed throughout the trial, or is hidden once text entry begins. Optionally, participants can be presented with their error rates after each trial, after each block of trials, or not at all. An easy-to-parse data file is produced.

The `Analyse` class reads the data file (or a collection of data files) and calculates the various statistics described in this paper and in [10, 11]. Care was taken in designing the data file format. Our intention is that even if researchers wish to create custom experiment software, the `Analyse` should still be useful for analysing their data.

CONCLUSIONS

This paper extends previous work by introducing two new performance metrics. The availability of a new software package is announced, providing an implementation of this text-entry error-rate analysis methodology that will be of interest to researchers and students.

ACKNOWLEDGMENTS

We thank Jacob Wobbrock for his e-mail discussions concerning the *Corrected And Wrong* and *Corrected But Right* error rates. This research was funded by NSERC.

REFERENCES

1. Card, S. K., Moran, T. P., & Newell, A. The keystroke-level model for user performance time with interactive systems. *Communications of the ACM*. ACM Press (1980). 23(7), 396-410.
2. eTForecasts, http://www.etforecasts.com/products/ES_pdas2003.htm (June, 2003)
3. GSM World. <http://www.gsmworld.com/news/statistics/index.shtml> (2003)
4. MacKenzie, I. S. Introduction to this special issue on text entry for mobile computing. *Human-Computer Interaction*. Lawrence Erlbaum Associates (2002). 17, 141-145.
5. MacKenzie, I. S. KSPC (keystrokes per character) as a characteristic of text entry techniques. *Proceedings of the Fourth International Symposium on Human-Computer Interaction with Mobile Devices*. Springer-Verlag (2002). 195-210.
6. MacKenzie, I. S., & Soukoreff, R. W. A character-level error analysis technique for evaluating text entry methods. *Proceedings of the Second Nordic Conference on Human-Computer Interaction – NordiCHI 2002*, ACM Press (2002). 241-244.
7. MacKenzie, I. S., & Soukoreff, R. W. Text entry for mobile computing: Models and methods, theory and practice. *Human-Computer Interaction*. Lawrence Erlbaum Associates (2002). 17, 147-198.
8. MacKenzie, I. S., & Soukoreff, R. W. Phrase sets for evaluating text entry techniques. *Extended Abstracts of the ACM Conference on Human Factors in Computing Systems – CHI 2003*. ACM Press (2003). 754-755.
9. Pavlovych, A., & Stuerzlinger, W. Less-Tap: A fast and easy-to-learn text input technique for phones, *Proceedings of Graphics Interface – GI 2003*. Canadian Information Processing Society (2003). 97-104.
10. Soukoreff, R. W., & MacKenzie, I. S. Measuring errors in text entry tasks: An application of the Levenshtein string distance statistic. *Extended Abstracts of the ACM Conference on Human Factors in Computing System – CHI 2001*. ACM Press (2001). 319-320.
11. Soukoreff, R. W., & MacKenzie, I. S. Metrics for text entry research: An evaluation of MSD and KSPC, and a new unified error metric. *Proceedings of the ACM Conference on Human Factors in Computing Systems – CHI 2003*. ACM Press (2003). 113-120.
12. Soukoreff, R. W., & MacKenzie, I. S. Input-based language modelling in the design of high performance input systems. *Proceedings of Graphics Interface – GI 2003*. Canadian Information Processing Society (2003). 89-96.
13. Wobbrock, Jacob O. Personal correspondence. (2003)
14. Yamada, H. A historical study of typewriters and typing methods, from the position of planning Japanese parallels. *Journal of Information Processing*. Information Processing Society of Japan (1980). 2(4),175-202.