Input Observer: Measuring Text Entry and Pointing Performance from Naturalistic Everyday Computer Use

Abigail Evans[1]
Jacob O. Wobbrock[1]

[1]The Information School
DUB Group
University of Washington
{abievans, wobbrock}
@uw.edu

CHI 2011, May 7–12, 2011, Vancouver, BC, Canada.
ACM 978-1-4503-0268-5/11/05.

Abstract
In this paper we describe the Input Observer, a background application that will be capable of measuring a user's text entry and pointing abilities from everyday computer use "in the wild." The application runs quietly in the background of the user's computer and utilizes global Windows Hooks to observe the text entry input stream and use of the mouse, and will yield data equivalent to results from lab-based measures of text entry and target acquisition. A major challenge is the lack of a task model from which researchers can know the *intent* of the user at every moment. We describe our approach to handling this issue for both text entry and mouse pointing.

Keywords
Text entry, mouse pointing, field studies, naturalistic computer use.

ACM Classification Keywords
H.5.2 [Information Interfaces and Presentation]: User interfaces—*Evaluation/methodology*.

General Terms
Human Factors, Measurement.

Introduction

People with motor impairments face numerous challenges when using computers. They either need to adapt themselves to hardware and software that has not been designed with the needs of motor-impaired users in mind, or they need to use assistive technology designed for users with a range of motor impairments that may still fall short of addressing the needs of the individual user.

Previous work by Gajos *et al.* [6] has shown that it is possible to create highly accessible computer interfaces in which the interface adapts to fit the abilities of the user. In order to build software that is better suited to the abilities of individual users, however, designers and software engineers need to be able to measure and evaluate those abilities, ideally in naturalistic settings from everyday computer use. But currently, such evaluations are costly and time-consuming.

Most interactions with a computer involve text entry, pointing, and target acquisition. There are many ways to test these skills in lab settings but no comprehensive way to measure them in the field. As lab studies can be difficult for users with disabilities [2,5], the ability to quantify text entry, pointing, and target acquisition in the field is especially important when considering the experience of users with impairments.

Building on existing methods for measuring text entry, and mouse movements in the lab, we are developing the *Input Observer* to measure those skills in the field. The application runs on Windows and uses global Windows Hooks to observe low-level text entry input streams and mouse movements. The application has two components: the *Text Observer*, which measures text entry speed and error rates; and the *Mouse Observer*, which measures target acquisition and pointing accuracy. The Input Observer runs quietly in the background of a user's computer, gathering data with minimal disruption.

At the time of writing, the Text Observer is close to completion while the Mouse Observer is still in its formative stages. This paper details the work done so far, some of the challenges encountered in moving from the lab to the field, and the next steps as we continue to develop the Input Observer.

Text Entry Speed

In lab tests of text entry speed, participants are shown a phrase which they transcribe as quickly and accurately as possible. The transcribed phrase represents a trial, which is then used to calculate text entry speed as words per minute [12].

In the field, however, there is no phrase presented for participants to transcribe so it is difficult to determine what should constitute a trial. The most obvious place to segment entered text to form a trial is when the user types a period, as this is likely, but not always, the end of a sentence and therefore would be similar to the phrases used in lab tests. Although abbreviations are not explicitly handled by our current implementation, they may result in trials that are too short and are therefore discarded. Other typical end-of-sentence characters, such as the exclamation mark, question mark, and the enter key, also can signify the end of trial, as does the use of the mouse.

With no presented phrase, users will be composing as they type, pausing to think and making edits as they go

along. The first challenge is to distinguish between pauses relevant to text entry speed, and pauses that represent thinking time or result from an external distraction. As it is not possible to determine a user's intention when there is a pause in text entry, we decided to end a trial after a pause of 300 ms in order to prevent irrelevant pauses from distorting results. The figure of 300 ms was arrived at through trial-and-error. We recognize that the difference between a pause due to text entry speed and a pause due to thinking time or other distractions is likely to vary among participants. For this reason, the user interface allows the pause length to be changed. This setting will be tested once the Input Observer is deployed.

Determining a Text Entry "Trial"

While the time taken to make and correct errors (mistakes or typos) should be included in the calculation of words per minute, edits (*i.e.*, word changes) should not be included. For example, if the user first types "See Spot rin", and immediately corrects "rin" to "run", the time taken to change the "i" to a "u" should be included in the duration of the trial. However, if the user were then to delete "run" and replace it with "walk", it would be considered an edit and neither the time taken to make the change, nor the change in wording, should be included in the trial as it would distort error rate calculations. Thus, we must distinguish edits from corrections.

Without a presented phrase to compare to the phrase entered by the user, it is difficult to determine the user's intention when they make changes to entered text. We decided that if the user backspaces through an entire word – from the last character entered up to or through a space – it is counted as an edit and the trial is ended at the last character entered that has not been deleted. If the user backspaces through a partial word, it is counted as a correction and the trial continues, including the backspaces and erased characters.

Although this method of distinguishing between edits and corrections is not perfect, we have had very encouraging results for calculation of words per minute during development. So far, the average words per minute given by the Text Observer after 30-50 trials, as tested by the authors, has consistently been within 5% of the result obtained in a conventional lab test. Although the Text Observer will dismiss some error corrections as edits and will count some edits as error corrections, it seems to be accurate in most cases. We will have to pay close attention to this aspect of speed calculation when the software is deployed.

Text Entry Error Rates

Calculating text entry speed in naturalistic computer use has been fairly straightforward. Calculating text entry error rates, however, is more difficult.

Soukoreff and MacKenzie [11] describe two types of errors in text entry: Incorrect Fixed, or corrected errors, and Incorrect Not Fixed, or uncorrected errors. Corrected errors are easily identified, as described above, when the user backspaces through one or more characters within a word. Uncorrected errors, on the other hand, require more work to identify because the user does not take any action that signals a possible error, and with the Input Observer, we only have access to the user's low-level actions. The first step in tackling this problem was to include a lexicon for error checking in the Input Observer. Once the user has entered a complete trial, the entered text is broken into

words using spaces and appropriate punctuation as delimiters. Each word is then checked against the lexicon, which contains nearly 80,000 words and is derived from the data freely available from the Washington University in St. Louis English Lexicon Project [4]. If the word is found in the lexicon, no further checking is needed. If, however, the word is not found, more steps need to be taken to confirm that it contains an error, and also to identify the most likely correct spelling of the word in order to determine exactly how many errors are present.

Web search engines find occurrences of words that are unlikely to appear in the lexicon but are correctly entered - commonly used abbreviations or proper nouns, for example. When a word is not found in the lexicon, the Text Observer searches for the word on Google using the "define:" command. For example, if the user enters "freind", it will not be found in the lexicon, so the Text Observer searches for the term "define:freind". If Google returns a list of definitions the word is deemed correct. If, however, the word is not defined, the search will return a statement saying that no definitions were found. Google will often offer an alternative spelling, for example, "Did you mean: friend?" This alternative word is taken to be the correct word. The Text Observer then uses the algorithm given by Mackenzie and Soukoreff [10] to calculate the minimum string distance between the "correct" word suggested by Google and the word entered by the user. To reduce the number of future queries to Google, the suggested word is added to a list of "learned" words, which is used for error checking alongside the lexicon.

If no definitions are found for a word entered by the user and Google cannot offer any alternative spellings, it is not possible to calculate the minimum string distance for that word. In that case, the average minimum string distance is calculated and any erroneous words that could not be resolved by Google are assumed to exhibit that same average error rate.

Initially, whenever a segmenting event occurred, the text entered since the previous segmenting event was considered a trial and words per minute and error rates would be determined for that trial. This meant that very short bursts of text entry could be counted as a trial. Although this did not greatly affect the measurement of text entry speed, there were some concerns about the effect on the uncorrected error rate. There are numerous cases where a user may enter a short string of characters that would look like an error to the Input Observer despite being accurate. For example, website addresses and search terms auto-completed after a few characters could look like errors. To address this issue, a text segment now has to be at least a minimum length of 24 characters in order to qualify as a trial. Twenty-four characters is one standard deviation less than the mean length of phrases in the Mackenzie and Soukoreff [9] phrase set. As that is the phrase set used in the lab test that will provide the baseline data to be compared with the Text Observer results, it was the most relevant corpus from which to calculate a minimum trial length.

The minimum trial length also addresses a privacy concern. Trial data is recorded for error checking but, as usernames and passwords are likely to be shorter than 24 characters, the minimum trial length prevents such data being recorded. However, raw data long enough to qualify as a trial is logged to an XML file in readable format. We are using the log to refine the Text

Observer but in the released version logging will be disabled unless the user decides to enable it. This should prevent personal information from being recorded without consent.

Another issue potentially leading to accurately entered text appearing incorrect occurs when a segmenting event arises in the middle of a word. For example, if a user starts to type a word and pauses before continuing, the text entry input stream will be segmented at that point, leaving a partial word. In such cases, the word will be split between two trials and each half of the word will be checked for errors independently. We addressed this problem by identifying trials for which the first character was immediately preceded by a letter or hyphen, as these characters indicate that the first word in the trial may actually be a section of a longer word. Next, the word is extended to include the section of the word that was entered before the trial began and the complete word is checked for errors (the extended first word is not used in the calculation of text entry speed). Any errors found are included in the second of the two adjacent trials to ensure the partial words are not counted twice.

Testing so far has shown the average errors given by the Text Observer for both types of errors to be within 2% of lab test results. However, the error rates given by the Text Observer are also consistently lower than those from the lab test. The cause of this effect seems to be the use of search engines for error checking. Many incorrect words, whether typos or word fragments, have definitions on Google, potentially causing some errors to be considered correct. The lab-tested error rates being used as baselines during development are low (less than 2%) so it remains to be seen how the error rates calculated by the Input Observer will hold up with users who produce higher error rates.

The Mouse Observer

While the Text Observer is almost ready for testing with users in the field, we are still brainstorming how to measure a user's mouse pointing skills in the field. Previous research [1] has assessed the validity of Fitt's Law in the field. We intend to go a step further by computing Fitt's Law throughput [8] and error rates. The first step is to figure out how to determine a single pointing attempt, *i.e.*, the pointing "trial." Although a pointing attempt will usually end with a click of the mouse, determining the start of the pointing attempt is less clear. A user may move the mouse around without clicking on anything so a single pointing attempt cannot be assumed to be all cursor movements between one click and the next. We will most likely need to identify a suitable length for a pause between movements. Also, we can smooth the raw velocity profile and then look for an initial ballistic movement followed by corrective submovements, which would indicate the expected profile for a single pointing event. From this profile information, the Mouse Observer can calculate the distance and timespan of each movement

The Mouse Observer will also need to determine the dimensions of the target the user clicks upon. With about 74% of screen elements, we can get those dimensions from the Windows Accessibility API [7]. However, many elements will be harder to access. Eventually, *Prefab* [3] may make getting screen target sizes possible just from targets' drawn pixels. For now, however, a different approach is required.

To determine pointing errors, the Mouse Observer will not only need to recognize when the user has missed a target, it will also need to get the dimensions of the intended target. For now, this task may be best suited to human judgment. Our intention is to scrape pixels in the vicinity of each mouse click, upload those as thumbnails to Mechanical Turk, and provide turkers the means to indicate whether a click was a hit or miss, and the dimensions of the target. While automated approaches have been tried [7], ours will be the first to crowdsource this determination.

Conclusion

The Input Observer will be able to collect data on a user's text entry and mouse input in a single deployment in the field. The measures given by the Input Observer will include text entry speed, corrected and uncorrected text entry error rates, pointing speed and accuracy, and Fitts' law pointing performance measures and models. The measures will be comparable to those that could be obtained in lab tests of text entry and pointing performance. As the software runs in the background and does not require any special attention from the user, it will be convenient and easy to use. The Input Observer will be a valuable to tool to researchers wanting to validate innovations or therapists evaluating interventions in the naturalistic everyday computer use.

Acknowledgements

This work was supported by the National Science Foundation under grant IIS-0952786.

References

[1] Chapuis, O., Blanch, R. and Beaudouin-Lafon, M. (2007). Fitts' law in the wild: A field study of aimed movements. *LRI Technical Repport Number 1480*. Laboratorie de Recherche en Informatique. Orsay, France: Universite de Paris Sud.

[2] Coyne, K.P. Conducting simple usability studies with users with disabilities. *Proc. HCI Int'l 2005*. Lawrence Erlbaum Associates (2005).

[3] Dixon, M. and Fogarty, J. A. (2010). Prefab: Implementing advanced behaviors using pixel-based reverse engineering of interface structure. In *Proc. CHI 2010*. New York: ACM Press, 1525-1534.

[4] English Lexicon Project. http://elexicon.wustl.edu/

[5] Feng, J., Sears, A. and Law, C.M. Conducting empirical experiments involving participants with spinal cord injuries. *Proc. HCI Int'l '05*. Lawrence Erlbaum Associates (2005).

[6] Gajos, K.Z., Wobbrock, J.O. and Weld, D.S. Improving the performance of motor-impaired users with automatically-generated, ability-based interfaces. *Proc. CHI 2008*. ACM Press (2008), 1257-1266.

[7] Hurst, A., Hudson, S. E. and Mankoff, J. (2010). Automatically identifying targets users interact with during real world tasks. *Proc. IUI 2010*. New York: ACM Press, 11-20.

[8] MacKenzie, I.S. Fitts' law as a research and design tool in human-computer interaction. *Human-Computer Interaction 7,* 1 (1992), 91-139.

[9] MacKenzie, I.S. and Soukoreff, R.W. Phrase sets for evaluating text entry techniques. In *Extended Abstracts CHI 2003*. ACM (2003), 754-755.

[10] Soukoreff, R.W. and MacKenzie, I.S. Measuring errors in text entry tasks: An application of the Levenshtein string distance statistic. *Extended Abstracts CHI 2001*. ACM Press (2001), 319-320.

[11] Soukoreff, R.W. and MacKenzie, I.S. Recent developments in text-entry error rate measurement. *Extended Abstracts CHI 2004*. ACM Press (2004), 1425-1428.

[12] Wobbrock, J.O. Measures of text entry performance. In *Text Entry Systems: Mobility, Accessibility, Universality*, I. S. MacKenzie and K. Tanaka-Ishii (eds.). Morgan Kaufmann, San Francisco, 47-74, 2007.