TrueKeys: Identifying and Correcting Typing Errors for People with Motor Impairments

Shaun K. Kane,[1] Jacob O. Wobbrock,[1] Mark Harniss,[2] Kurt L. Johnson[2]

[1]The Information School
University of Washington
Box 352840
Seattle, WA 98195 USA
{skane, wobbrock}@u.washington.edu

[2]Department of Rehabilitation Medicine
University of Washington
Box 357920
Seattle, WA 98195 USA
{mharniss, kjohnson}@u.washington.edu

ABSTRACT

People with motor impairments often have difficulty typing using desktop keyboards. We developed *TrueKeys*, a system that combines models of word frequency, keyboard layout, and typing error patterns to automatically identify and correct typing mistakes. In this paper, we describe the TrueKeys algorithm, compare its performance to existing correction algorithms, and report on a study of TrueKeys with 9 motor-impaired and 9 non-impaired participants. Running in non-interactive mode, TrueKeys performed more corrections than popular commercial and open source spell checkers. Used interactively, both motor-impaired and non-impaired users performed typing tasks significantly more accurately with TrueKeys than without. However, typing speed was reduced while TrueKeys was enabled.

ACM Classification Keywords

H.5.2. Information interfaces and presentation: User interfaces — *Input devices and strategies.*

Author Keywords

Computer access, motor impairments, typing errors, minimum string distance, spell checking, error correction.

INTRODUCTION

Correcting typing errors is one of the most common activities performed by computer users. However, users with motor impairments may produce significantly more typing errors than other users, and may require more time to correct these errors [10]. Assistive technologies such as keyguards and word prediction software may reduce typing errors, but may also decrease typing speed [4,7], and are often expensive or difficult to learn. For these reasons, many users with motor impairments avoid specialized input devices and instead use standard keyboards [10].

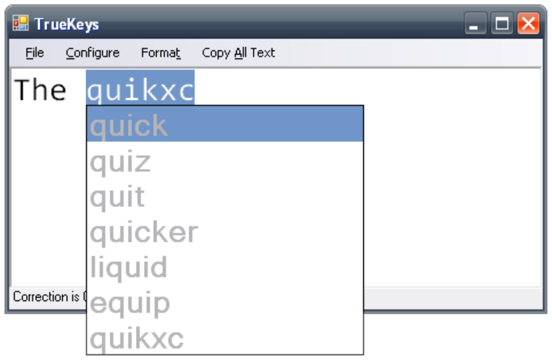

Figure 1. The TrueKeys user interface performing a correction from "quicxc" to "quick".

To address these issues, we developed *TrueKeys*, a system that automatically corrects typing errors. TrueKeys employs models of word frequency, keyboard layout, and typing error patterns to identify and correct typing mistakes. Used non-interactively, the TrueKeys algorithm performs better than several common spell checkers. Used interactively, TrueKeys significantly increases typing accuracy for both motor-impaired and non-impaired users, although it can slow users down somewhat.

RELATED WORK

Some prior systems have attempted to reduce typing errors produced by users with motor impairments. Trewin [10] introduced a typing filter called *OverlapKeys* that automatically corrects errors in which the user strikes two keys at once. VITIPI [1] is a word prediction system that can correct errors as the user types. TrueKeys goes beyond systems such as OverlapKeys in that it can correct many error types. However, TrueKeys is not a word prediction system like VITIPI. Instead, TrueKeys allows users to type normally, while automatically correcting errors in-place.

THE DESIGN OF TRUEKEYS

TrueKeys consists of two components: (1) a word correction algorithm; and (2) a user interface that allows users to interactively correct text while typing.

Correction Algorithm

The TrueKeys correction algorithm identifies mistyped words and suggests potential corrections. A word is first checked against a large word list to determine whether it is known. If it is unknown, TrueKeys creates a list of likely correction candidates and ranks them using a word distance score. The candidate with the lowest distance score replaces the original input.

A *minimum string distance* (MSD) score, derived from the Levenshtein string distance [5], is used to calculate the distance between the user's input and a correction candidate. This score is based on four *edit operations:* substitution, insertion, deletion and transposition [2]. These edit operations correspond to common typing errors. *Substitution* errors occur when the user presses an incorrect key instead of the intended key. *Insertion* errors occur when the user presses an incorrect key in addition to the intended key. *Deletion* errors occur when the user fails to press a key. *Transposition* errors occur when the user types two keys in reverse order. The MSD score is the number of operations needed to transform one string into the other [8].

TrueKeys extends MSD with information about the physical layout of the keyboard to produce a weighted MSD score we call *wMSD*. As suggested by Deorowicz and Ciura [3], physical distance between keys is used as a weighting factor for edit operations. Table 1 describes how distance weights are used. wMSD is combined with two frequency scores: the frequency of the word alone (f_{word}), and the frequency of the word given the previous word (f_{bigram}), to produce the final distance score, $score_{TK}$ (Eq. 1).

$$score_{TK} = \alpha \cdot wMSD(S_{typed}, S_{word}) + \beta \cdot f_{word} + \gamma \cdot f_{bigram} \quad (1)$$

A lower $score_{TK}$ indicates a better match. The terms α, β, and γ are weighting factors derived from pilot data. The β and γ factors are negative so that common words produce a lower score. For the pilot study we used the following empirically determined weights: $\alpha = 0.4$, $\beta = -0.00001$ and $\gamma = -0.0001$. We used a weight of $\varepsilon = 1.5$ for deletion errors.

Substitution	Weighted by the distance between the entered and intended key (cost = distance * α).
Insertion	Weighted by the distance between the entered key and the previous or next key, whichever is closer (cost = distance * α).
Deletion	Deletions are less common than other errors and are weighted more (cost = ε).
Transposition	Not weighted (cost = 1).

Table 1. Distance weights used in the wMSD score.

Finally, TrueKeys contains two typing filters to correct common errors. A *run-on error filter* detects errors in which two words are entered without a space in between, such as "quickbrown", and separates them. An additional *simultaneous key press filter* detects simultaneous key presses and considers both possible candidate words. For example, if the user presses the "q" and "w" keys together and then types "as", the system will attempt to recognize both "qas" and "was", choosing the latter.

TrueKeys User Interface

The TrueKeys interface provides: (1) visual feedback of word corrections; and (2) a simple keyboard-based interface for choosing among correction alternatives. Correction begins when the user presses a delimiter key such as ENTER or SPACE. If the entered word is not known, the system chooses the best correction candidate and automatically replaces the original string with the corrected word. The system then underlines the word to show that it has been replaced. Figure 1 shows the TrueKeys interface.

If the user is satisfied with the correction, he or she may continue to enter text. If the system makes an incorrect replacement, the user may delete the word using the BACKSPACE key, or may use the arrow keys to select from a 6-item *N-best* list of alternative correction candidates, including the original unadjusted word (Figure 1). TrueKeys records users' corrections, and will not repeat a correction that the user has overridden.

PERFORMANCE OF THE TRUEKEYS ALGORITHM

To evaluate the TrueKeys correction algorithm, we ran TrueKeys on a set of 360 phrases collected from our pilot study. BACKSPACE was disabled in order to capture all errors in the input stream. We identified and tagged 171 mistyped words from this data set that were not valid English words and were thus correctable by TrueKeys.

Of the 171 invalid words, TrueKeys successfully corrected 99 (57.9%). An additional 31 (18.1%) words were not changed correctly, but contained the correct word in the 6-item list of correction candidates. The remaining 41 (24.0%) words were not corrected by TrueKeys. Seventeen words were typed correctly but were identified by TrueKeys as mistyped and were incorrectly changed. Thus, despite some false positives, TrueKeys had an overall positive effect (48.0%) on text accuracy.

We also compared the performance of the TrueKeys correction algorithm to Microsoft Word 2004 and the open source spell checkers *ispell* and *aspell.* Comparing TrueKeys to these systems allows us to judge the effectiveness of each algorithm at correcting typing errors. Figure 2 shows the number of words corrected by each algorithm from the list of 171 words. TrueKeys corrected more total words than each of the other algorithms.

USER EVALUATION

We performed a pilot user evaluation of TrueKeys with 9 motor-impaired and 9 non-impaired participants. All participants performed two tasks: a series of phrase transcription trials using phrases chosen randomly from a standard set [6], and a short paragraph transcription (~500 characters). Participants performed both tasks with correction enabled and correction disabled.

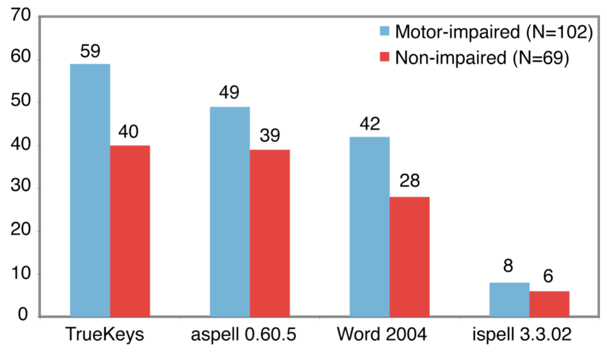

Figure 2. Mistyped words corrected by various spell checkers. TrueKeys corrected the most words. These are single measures and do not reflect an average or distribution.

Tasks were performed using an Apple MacBook laptop computer with a full-sized Dell USB keyboard. Two participants were unable to use the USB keyboard and instead used the smaller laptop keyboard, while another participant required *StickyKeys*. All other participants used standard keyboard settings. The *TextTest* application [12] was used to administer and log transcription tasks.

Phrase Transcription Task

Each participant performed 20 trials with TrueKeys enabled and 20 trials with TrueKeys disabled. A single trial involved the transcription of a single phrase (~30 characters). Phrases contained only letters. Participants were instructed to transcribe the phrase presented on the screen and to strive for both speed and accuracy.

Use of the BACKSPACE key was disabled for the phrase transcription task for two reasons. First, disabling BACKSPACE provided us with a stream of uncorrected typing data that enabled us to evaluate the correction algorithm without the influence of user correction. Second, disabling BACKSPACE encouraged participants to use TrueKeys rather than correct everything manually.

The order of the TrueKeys *Correction* (on, off) treatments was counterbalanced. There was neither a significant effect of *Order* on words per minute (WPM) ($F_{1,16}$=0.01, n.s.) nor a significant *Order*Correction* interaction ($F_{1,16}$=0.01, n.s.), indicating adequate counterbalancing.

Design and Analysis

The experiment was a mixed between- and within-subjects factorial design with the following factors and levels: (1) *Impairment* (motor-impaired, non-impaired); and (2) TrueKeys *Correction* (on, off). Our dependent measures are participants' average speeds (WPM) and uncorrected error rates [9] over trials.

The TrueKeys prototype allowed users to type over text while correcting. Because BACKSPACE was disabled for the phrase typing trials, this feature could provide an unfair advantage to TrueKeys. Therefore, we excluded from analysis all 30 trials in which a participant used this feature.

Error Rate

Uncorrected error rates are used to measure the accuracy of the final transcribed text [9]. Average values are shown in Figure 3a. Our data showed a significant main effect of *Impairment* ($F_{1,16}$=4.90, p<.05) and *Correction* ($F_{1,16}$=4.82, p<.05) on uncorrected errors, but no significant *Impairment*Correction* interaction ($F_{1,16}$=0.31, n.s.). Thus, TrueKeys correction does indeed reduce errors, and it does so about evenly for each subject group.

Speed

Words per minute (WPM) is used to measure speed. Average values are shown in Figure 3b. There were significant effects of *Impairment* ($F_{1,16}$=77.95, p<.0001) and *Correction* ($F_{1,16}$=17.27, p<.001) on speed, but no significant *Impairment*Correction* interaction ($F_{1,16}$=0.78, n.s.). Thus, correction reduced overall errors, but reduced entry speed similarly for both participant groups. We revisit this tradeoff in the discussion section.

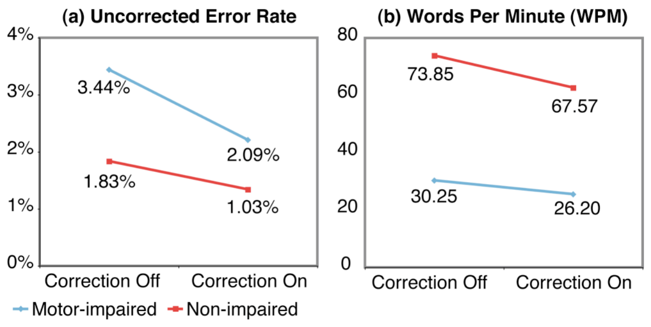

Figure 3. (a) TrueKeys significantly reduces error rate for both participant groups, but (b) also reduces speed.

Paragraph Transcription Task

Participants transcribed a 100-word passage that included capital letters and punctuation. However, the use of TrueKeys correction did not significantly affect speed or error rates for paragraph transcription. These statistics are therefore omitted from the current work. It is possible that the short length of the transcription paragraph prevented us from observing statistically significant differences in speed and accuracy.

User Acceptance of Corrections

During the paragraph task, TrueKeys corrected 46 words typed by the participants. Of these, 29 (63.0%) corrections were accepted by the user. An additional 7 (15.2%) corrections were changed by the user using the correction menu. The remaining 10 (21.7%) corrections were overwritten by the user. This indicates that participants generally accepted the corrections provided by TrueKeys.

Analysis of Typing Errors

In order to inform the further design of the TrueKeys correction algorithm, we examined the word-level errors produced during the study. Overall, users with motor impairments averaged a higher MSD per word (1.48, *SD*=0.89) than non-impaired users (1.10, *SD*=0.34). A Wilcoxon rank-sum test showed that this difference is significant ($\chi^2_{(1,N=176)}$=11.90, p<.001). Motor-impaired users also mistyped more words overall (11.67, *SD*=9.49) than non-impaired users (7.89, *SD*=4.40), but this difference is non-significant ($\chi^2_{(1,N=18)}$=0.64, n.s.).

The distribution of error types also varied across participant groups. Most notably, motor-impaired users performed more insertion errors than non-impaired users. Table 2 shows the distribution of errors for each participant group.

	Motor-impaired	**Non-impaired**
Insertions	113 (76.4%)	45 (60.0%)
Deletions	8 (5.4%)	12 (16.0%)
Substitutions	21 (14.2%)	13 (17.3%)
Transpositions	6 (4.1%)	5 (6.6%)

Table 2. Number of word-level errors for each participant group. Numbers in parentheses indicate the relative percentage of each error and sum to 100% for each group.

DISCUSSION
In the phrase transcription task, use of TrueKeys correction reduced entry speed somewhat. It is unclear whether these effects would diminish over time as users become more accustomed to the system. Several participants commented that using TrueKeys effectively would require them to unlearn established typing habits and develop new habits. One user stated, "It's hard to reprogram my brain and my typing. To use [TrueKeys] is to change decades of habit." Although TrueKeys provided significantly increased accuracy for both motor-impaired and non-impaired users, users with motor impairments seemed more interested in and more willing to use the system than non-impaired users. It is possible that inaccurate typists would be more willing to accept TrueKeys' current speed-accuracy tradeoff.

CONCLUSION AND FUTURE WORK
We have presented TrueKeys, a system that accurately detects and corrects typing errors. Our experiments show that the TrueKeys correction algorithm is more effective than other commonly used spell checkers at correcting physical typing errors. Used interactively for typing phrases, TrueKeys significantly reduces typing errors for both motor-impaired and non-impaired users. Use of TrueKeys also reduced speed somewhat, although this effect might change over time. A longitudinal study is needed to determine how users will adapt to TrueKeys and incorporate it into their everyday typing behavior.

Performance of TrueKeys could be improved further through changes to both the correction algorithm and the user interface. During the user study, we observed that users repeatedly made similar errors. For example, one participant frequently struck the semicolon key when attempting to use the right side of the keyboard. Including a personalized model of a user's typing errors might enable TrueKeys to more accurately correct these errors. The user interface might also be redesigned to be less intrusive. This might reduce the speed hit observed in the present study. One possibility is to explore different user interfaces for motor-impaired and non-impaired users, while using the same underlying algorithm for both groups.

Finally, while we have focused on desktop keyboards for this study, we believe that TrueKeys may be useful for correcting typing errors made on mobile device keyboards, and for typing when users are on the move.

ACKNOWLEDGMENTS
The authors thank Sheryl Burgstahler, Dan Comden and Curt Johnson for recruitment assistance, and Shari Trewin for providing the paragraph text used in the experiment.

REFERENCES

1. Boissiere, P. and Dours, D. (1996). VITIPI: Versatile interpretation of text input by persons with impairments. In *Proc. ICCHP '05*, R. Oldenbourg Verlag, 165-172.
2. Damerau, F.J. (1964). A technique for computer detection and correction of spelling errors. *Communications of the ACM*, *7 (3)*, 171-176.
3. Deorowicz, S. and Ciura, M.G. (2005). Correcting spelling errors by modelling their causes. *International Journal of Applied Mathematics and Computer Science*, *15 (2)*, 275-285.
4. Koester, H.H. and Levine, S.P. (1996). Effect of a word prediction feature on user performance. *Augmentative & Alternative Communication*, *12 (3)*, 155-168.
5. Levenshtein, V.I. (1966). Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, *10*, 707-710.
6. MacKenzie, I.S. and Soukoreff, R.W. (2003). Phrase sets for evaluating text entry techniques. In *Proc. CHI '03*, ACM Press, 754-755.
7. McCormack, D. (1990). The effects of keyguard use and pelvic positioning on typing speed and accuracy in a boy with cerebral palsy. *American Journal of Occupational Therapy*, *44 (4*), 312-315.
8. Soukoreff, R.W. and MacKenzie, I.S. (2001). Measuring errors in text entry tasks: an application of the Levenshtein string distance statistic. In *Proc. CHI '01*, ACM Press, 319-320.
9. Soukoreff, R.W. and MacKenzie, I.S. (2003). Metrics for text entry research: an evaluation of MSD and KSPC, and a new unified error metric. In *Proc. CHI '03*, ACM Press, 113-120.
10. Trewin, S. (2002). An invisible keyguard. In *Proc. Assets '02*, ACM Press, 143-149.
11. Wobbrock, J.O. and Myers, B.A. (2006). Analyzing the input stream for character-level errors in unconstrained text entry evaluations. *ACM Transactions on Computer-Human Interaction*, *13 (4)*, 458-489.