Evaluation of a Smart-Restorable Backspace Technique to Facilitate Text Entry Error Correction

Ahmed Sabbir Arif[1], Sunjun Kim[3], Wolfgang Stuerzlinger[2], Geehyuk Lee[3], Ali Mazalek[1]

[1]Ryerson University
Toronto, Ontario, Canada
{asarif, mazalek}@ryerson.ca

[2]Simon Fraser University
Surrey, British Columbia, Canada
http://ws.iat.sfu.ca

[3]KAIST
Daejeon, South Korea
{kuaa.net, geehyuk}@gmail.com

ABSTRACT

We present a new smart-restorable backspace technique to facilitate correction of "overlooked" errors on touchscreen-based tablets. We conducted an empirical study to compare the new backspace technique with the conventional one. Results of the study revealed that the new technique improves the overall text entry performance, both in terms of speed and operations per character, by significantly reducing error correction efforts. In addition, results showed that most users preferred the new technique to the one they use on their tablets, and found it easy to learn and use. Most of them also felt that it improved their overall text entry performance, thus wanted to keep using it.

Author Keywords

Text entry; virtual keyboard; predictive text; touchscreen; errors; error correction; backspace.

ACM Classification Keywords

H.5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous; H.5.2 User Interfaces (D.2.2, H.1.2, I.3.6): Input devices and strategies (e.g., mouse, touchscreen).

INTRODUCTION

Recent market research suggests that tablet computers will gradually overtake desktop and laptop computers in the near future [11]. Tablets have become lighter and smaller over the years, with substantial advances in power, functionality, and performance. Nowadays, one can be productive with tablets for tasks that previously required the "power" of a desktop or a laptop. Text entry with tablets, however, is still relatively difficult due of their smaller screen sizes and the absence of a physical keyboard. Although many have addressed these issues by developing new and improved text entry techniques, almost no work has focused on the error correction process. Most current techniques allow users to correct errors in three different ways. First, navigating to the position where a mistake was made by using direct touch cursor control, correcting the mistake by replacement, insertion, or deletion, and then repositioning the cursor to the end of the text. Second, using the virtual cursor control keys, which requires navigating to the error position by repetitively tapping on virtual cursor control keys (if they are available), correcting the mistake, and then repositioning the cursor to the end of the text. Finally, through the backspace key, which involves deleting all characters following the error position, correcting the mistake, and then re-inputting the deleted text.

While these methods work reasonably well for immediate error corrections, *i.e.*, when an error is noticed and corrected immediately after committing it or within the entry of three characters following the error, they all suffer from usability issues when users attempt to correct a missed or overlooked error in the text. In such a scenario, with direct cursor control, users are required to tap outside of the virtual keyboard, *i.e.*, on the text area, to precisely place the cursor to the intended position, correct the mistake, and then reposition the cursor to the end of the text. This increases the cognitive load and the index of difficulty (ID, a function of the target size and distance in Fitts' law [14]) for the task. Such an action has been shown to require on average 4.5 seconds [16]. Besides, positioning the cursor among small characters is error prone and increases the possibility of further errors [7].

Both backspace and cursor control keys are more precise and accurate than direct cursor control, but require potentially many repetitive keystrokes. For instance, to position the cursor five characters behind the current position, users have to tap the cursor-right key five times. Then after fixing the error, have to navigate back to the original position with five additional keystrokes. Thus ten plus the corrective keystrokes are required to fix a single error. Similarly, with the backspace key, users have to delete all text following the error, and after correcting it, have to re-input the deleted text, requiring the same number of keystrokes as the cursor control key method.

To facilitate such error correction episodes, we present here a new smart-restorable backspace technique for touchscreen-based tablets. The reminder of the paper is organized as follows. It starts with a brief discussion of related work in the area. Then, it introduces a basic version of the new approach, justifies the design decisions, and evaluates it in a pilot study that provides further insights into the human error correction process and indications on how to improve the technique. The paper then presents an improved approach, compares it with the conventional backspace method in the main user

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
CHI'16, May 07-12, 2016, San Jose, CA, USA
© 2016 ACM. ISBN 978-1-4503-3362-7/16/05$15.00
DOI: http://dx.doi.org/10.1145/2858036.2858407

study, and discusses the results. Finally, it ends with a conclusion and speculations on future work.

RELATED WORK

Previous research has focused on touchscreen text editing [16], where users modify on an already completed text, and error prevention [3, 10, 12, 18], where the system attempts to prevent an error from occurring. Yet, not much work has focused on error correction itself, where users correct their mistakes as they type. In fact, the only major development in the area is a feature that is now default in almost all virtual predictive keyboards, which enables users to restore the originally inputted text by pressing the backspace key immediately after an autocorrection [11].

A recent mobile keyboard, KeuKey [24], permits users to correct typos by entering the correct word at the end of the text, and performing a down gesture on the keyboard to replace the incorrect word with the correct one. It simply compares the last inputted word with the rest of the text, and replaces the word with the best match score. If multiple words match, it replaces the most recent word, but permits the user to navigate to the correct word with additional down gestures. With this technique, users are often forced to enter a complete word even when they only made a single mistake. Fleksy [15] allows users to delete text word by word by performing left gestures on the keyboard. While this permits users to quickly reach an error position, it still requires them to delete the correct portion of the text and to re-input it later.

The new Apple iOS 9 provides support for indirect cursor control using two-finger gestures [1] and 3D touch [2] on the virtual keyboard. While this enables users to move the cursor in any direction by words or lines, prior investigations have found such approach less accurate than direct touch [27].

THE SMART-RESTORABLE BACKSPACE

The smart-restorable backspace utilizes the existing predictive system of a virtual keyboard to accommodate the correction of overlooked spelling mistakes in a text entry episode. It does not account for in-vocabulary (IV) errors, such as an incorrect autocorrection. While spelling mistakes and typos are less frequent in most current predictive techniques due to forceful autocorrection, they are mostly independent of the predictive method. Yet they still impact the user experience. The goal of our work is to enable users to correct such mistakes with fewer keystrokes than the default backspace. In this section, we discuss the design and implementation of our technique.

Determining Correction Positions

When the predictive system finds a likely misspelled word, the smart-restorable backspace compares it with the most probable correct word to find the position(s) where a single-character operation, *i.e.*, insertion, deletion, or substitution, is necessary to change the incorrect word to the correct one; referred to as correction positions (*CP*). The system uses the Levenshtein distance [25] and a string comparison algorithm [20] to determine these *CPs*. Then, it records the earliest correction position for future smart deletion. For example, if the user enters "*exapla*" instead of "*example*", the system will detect two potential correction positions, an insertion after the third character (*CP* 3), and the replacement of the last character (*CP* 5), but will record only the earliest one (*CP* 3).

Basic Smart Deletion

The basic smart-restorable backspace method behaves like the conventional backspace for the first three repetitive taps to accommodate immediate error corrections. For instance, when users notice a mistake almost immediately after committing it, *i.e.*, within the entry of three characters following the mistake, they can quickly make the necessary correction and continue with text entry. The system automatically triggers smart deletion only on the fourth backspace tap, provided that the text contains at least one misspelled word. In such a case, the technique deletes all input following the latest correction position. For example, if the user inputs the phrase, "*exaple has more folloes than reason*" instead of "*example has more followers than reason*", the first three backspaces will delete the last three characters ("*son*") but the fourth one will delete all input after the correction position for the latest misspelled word, *i.e.*, "*folloes*" instead of "*follows*", *CP* 21, and the user will then see "*exaple has more follo*".

In order to correct multiple misspelled words in the text, the user has to keep tapping on the backspace key. Then on each fourth tap the system will delete all input following the previous correction position. In the above example, the user can continue tapping on the backspace key to delete the text after the first correction position, *i.e.*, "*exapla*" instead of "*example*", corresponding to *CP* 3, "*exa*". This enables the user to delete 32 characters with only 8 backspace taps.

The system also accounts for direct cursor positioning. If the user moves the cursor to before the last misspelled word, the system disregards all correction positions following the cursor. *E.g.*, if the user moves the cursor before the last misspelled word, *i.e.*, "*folloes*" in the above example, the technique will disregard "*folloes*" and only account for the mistakes that precede the cursors, *i.e.*, "*exapla*".

The system also permits users to delete multiple characters by long-pressing the backspace key. When the user holds the backspace key for more than 750 milliseconds, the system starts deleting one character every 150 milliseconds, until the user lifts his/her finger. These thresholds are commonly used in mobile and Web interfaces.

Smart Restoration

One issue with smart deletion is that after deleting a chunk of text, users then have to re-input the correct portions. Our restoration feature builds on Kim and Lee's idea [23], and addresses this by keeping a separate record of the deleted text and later suggesting the correct remainder of it in the prediction panel when the user attempts to re-input it, *i.e.*, when the user inputs at least two characters which match previously entered text following the correction. If the deleted text contains errors, the system trims the text based

on each correction position to avoid the entry of incorrect text. For instance, in the above example, let us assume that after deleting all text until the first correction position, "*exa*", the user corrects the spelling of "*example*", and then enters a space and two characters of the next word, "*example ha*". In this case, the system will find a match for "*ha*" in the deletion record "*has more folloes than reason*", but will trim it to avoid the entry of a likely misspelled word "*folloes*" to display only the correct portion "*s more follo*" in the prediction panel. The user then can restore this text by tapping on it in the prediction panel. The system removes a *CP* and a deleted chunk from the record as soon as the user corrects a spelling and/or restores a chunk of text, to avoid recurrences.

The system uses the characters entered after the correction of a misspelled word, *i.e.*, two characters or more, to find a match in the deleted text to suggest a smart restoration. For instance, in the above example, when the user enters "*ha*" after correcting the misspelled word "*example*", the system searches the record for "*ha*". As there is only one occurrence of the key in the deleted text, the system simply suggests the restoration of the reminder of the text. However, in case of multiple occurrences of the search term, the system uses the Levenshtein distance [25] to find the best match. For instance, if "*she had coffee*" was also in the record, the system would have found two matches for "*ha*", one in "*le has more folloes than reason*" and another in "*she had coffee*". Nevertheless, as the recently entered text "*mple ha*" would yield a lower Levenshtein distance for "*le ha*" (distance: 2) than "*she ha*" (distance: 3), the former text would have been suggested for restoration. Figure 1 illustrates an example error correction episode with the smart-restorable backspace.

Word Prediction and Autocorrection

We developed a simple word prediction and autocorrection system using the most frequent 10,000 words in the English language [30]. Similar to the default Android predictive system, it displays a panel above the virtual keyboard to suggest three candidates: the original text in the left, the most probable prediction in the center (highlighted with underline), and the second most probable one in the right. Illustrated in Figure 1. The system permits users to enter a candidate by pressing the space key, which inputs the highlighted candidate, or by tapping on the candidate word in the panel.

For a likely misspelled word, the system autocorrects it with the most probable correct word, *i.e.*, the underlined candidate in the prediction panel. However, to replicate the default Android keyboard with its "aggressive" autocorrection, both of the following must hold to trigger an autocorrection.

- The length difference between the entered and the candidate word is less than two characters. That is, the inputted word is of the same length or one character longer or shorter than the candidate word.
- The Levenshtein distance [25] between the inputted and the candidate word is less than two operations.

If the above parameters are not met, the system highlights the likely misspelled word with a red underline. See Figure 1 (a). Users can still override an autocorrection by tapping on the original text in the panel, *i.e.*, the left candidate. Then, the system does not consider the word as misspelled and does not underline it. Users can also replace an autocorrected word with the original text by tapping on the backspace key directly after the autocorrection. Then the system does not autocorrect that word on the second input or correction attempt.

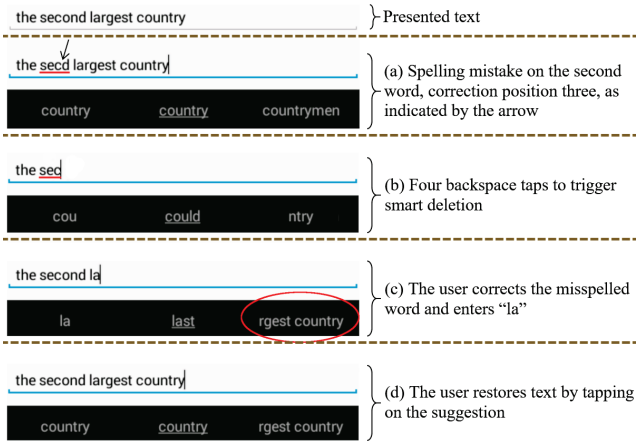

Figure 1. Four steps of an example error correction episode with smart backspace and restoration: (a) the user makes a spelling mistake on the second word, (b) taps backspace four times to delete the text after the latest correction position, (c) corrects the misspelled word and then enters two characters "la", which displays a restoration suggestion on the right of the prediction panel, then (d) restores previously deleted text by tapping on the suggestion.

We informally tested the custom predictive system with some experienced Android OS users. None of them noticed potential differences between the custom and the default Android prediction.

STUDY SETUP

To increase the external validity of the user studies evaluating the new smart-restorable backspace technique, we conducted an informal survey to identify the most used text entry positions and predictive features on touchscreen-based tablets. The survey involved 24 experienced users who owned and frequently used tablets for text entry. Ages were from 18 to 38 years, average 23.5 (SD = 5.7). 62.5% of them were female. Results revealed that about 92% of them use word prediction and 79% use autocorrection.

Almost all users input text by placing the device on flat surfaces, such as a table, and input text using their index fingers (54%), thumbs (29%), or all fingers but thumbs (8%). The remaining 9% hold the device with their hands and input text using their thumbs. Based on this, we placed the device on a table during the studies but allowed users to input text using their preferred posture. Similarly, as the survey showed that almost all users use word prediction and autocorrection, we enabled these features.

In another pilot, we evaluated an earlier version of the backspace technique *without* prediction. Three novices (one female), average age 26 years (SD=3.54), entered 70 short English phrases [26] with each technique, in counterbalanced order, on a first generation Apple iPad. Results revealed that the smart-restorable backspace increased text entry speed by 32.5% and reduced total error rate by 57.4%. This suggests the new technique substantially improves entry performance for novices when word prediction and autocorrection features are *disabled*. Yet, this is not ecologically valid and we re-enabled these features after this pilot.

A PILOT STUDY

We evaluated the basic smart-restorable backspace technique in a pilot study. The purpose was to fine-tune the technique's performance, to get user feedback, and also to guide the final study design [8].

Apparatus

We used a custom application developed with the Android SDK on an ASUS MeMO Pad HD 7 at 800×1280, 196.8×120.6×10.8 mm. The device used the default Google keyboard and logged all interactions. See Figure 1 (a). The device was attached to a desk with a custom dock to tilt it to a comfortable typing position at ~15°. See Figure 1 (b).

Participants

Twelve participants, aged from 19 to 38 years, average 23, participated in the pilot. Seven of them were female and one was left-handed. They all owned a tablet for at least a year and frequently used it for text entry.

(a) (b)

Figure 2. (a) The device and the custom application used during the pilot study and (b) a participant inputting text using the custom application during the pilot study.

Procedure and Design

We used a within-subjects design. The two factors were the conventional and the smart-restorable backspace. During the study, participants were asked to enter short English phrases from the MacKenzie and Soukoreff set [26] with both techniques. These phrases were shown in random order on the display, all in lowercase. We instructed each participant to read and understand the phrases in advance, then to enter them as quickly and accurately as possible. When finished, they had to press the enter key to see the next phrase. Error correction was *forced*. That is, participants had to correct all mistakes before submitting a phrase, otherwise the system made a "ding" noise to notify them of any existing error and did not accept the submission. The reasons behind this is that we observed that some users simply ignore some or all of the errors. The cause of this may be that current correction efforts are too high (a core motivation for our work). While sending erroneous text may be acceptable in some situations, this is not desirable in all contexts. To address the potential confound, we decided to force users to correct all errors.

Participants could use any comfortable posture in landscape position to input text. During the study, most of them (92%) used both hands. We provided them with two practice phrases before each condition to make sure that they were moderately familiar with the techniques and protocol. They could extend this practice period on request. They were informed that they could rest between conditions, blocks, or trials. Timing started from the entry of the first character and ended with the last. In summary, the design was:

12 participants ×
2 conditions (*conventional* and *new*, *counterbalanced*) ×
36 short phrases (excluding practice phrases)
= 864 submissions, in total.

Metrics

We calculated the following metrics during the study.

- **Words per minute (wpm):** the total number of words entered in a minute. This metric measures text entry speed.
- **Operations per character:** denotes the average number of operations it requires to input one character. This metric is similar to keystrokes per character (KSPC), but considers all operations, including taps and gestures.
- **Backspace rate (%):** signifies the average percentage of backspace action necessary to produce error free text. This was calculated as the ratio of the total number of backspace keys used to the length of the transcribed text.

Results

We used repeated-measures ANOVA for all analysis.

An ANOVA failed to find a significant effect of technique on entry speed ($F_{1,11} = 0.58$, ns). The average entry speed with the conventional and new backspace were 23.5 (SE = 1.55) and 22.68 wpm (SE = 1.5), respectively. We filtered the data for cases where users committed at least one spelling mistake (64% of all transcribed phrases), but failed again to identify an effect ($F_{1,11} = 0.04$, ns).

An ANOVA failed to find a significant effect of technique on operations per character ($F_{1,11} = 1.57$, ns). On average the conventional and smart-restorable backspace required 1.2 (SE = 0.05) and 1.2 (SE = 0.04) operations per character, respectively. There was no significant effect for cases where users committed at least one mistake ($F_{1,11} = 1.26$, ns).

There was no significant effect of technique on backspace rate ($F_{1,11} = 3.51$, ns). The average backspace rate for the conventional and smart-restorable backspace were 13.58 (SE = 2.22) and 10.72% (SE = 1.41), respectively. There was no significant effect for cases where users committed at least one mistake ($F_{1,11} = 4.67$, $p = .05$).

Discussion

Due to the inclusion of predictive features, we did not expect to observe a significant effect of technique on entry speed,

accuracy, and backspace rate. Results revealed that although the backspace rate was about 21% less with the new smart-restorable backspace, both techniques needed about the same number of operations per character. This suggests that despite using fewer backspaces than the conventional technique, users needed additional operations. More surprisingly, results showed that the entry rate with smart-restorable was 3.5% less. Through more analysis, we found several reasons for this.

First, participants had (too) few occasions to use the smart-restorable features when transcribing short English phrases. In an entire text entry episode, there were only about 29% cases where they *could* use the smart-restorable backspace. This corresponds to a previous finding [6], which showed that about 79% of all errors are noticed and then corrected within the entry of three characters following the mistake. This was also apparent in the post-study interviews, where many users commented that they did not notice any difference between the conventional and the smart-restorable backspace.

Second, the smart deletion feature disrupted the natural flow of text entry. It deleted chunks of text automatically, and often unexpectedly, not allowing users to mentally prepare for the change. As a result, the entry of a character following a smart deletion required more time. According to our logs users took on average 2.1 seconds to continue with text entry after smart deletion was triggered.

Third, the restore feature was rarely used by participants; most probably due to inadequate visual feedback. The system displayed all restorable chunks in the prediction panel, which required users to shift their focus from the text area to the panel. Besides, the restore feature did not record texts deleted with regular or tap-hold backspacing, *i.e.*, *only* with smart backspace, which caused confusion.

IMPROVED SMART-RESTORABLE BACKSPACE

Based on the findings of the above pilots and Kim and Lee's early exploration [23], we made several improvements to the smart-restorable backspace behavior.

First, we provide the users with more control over the smart deletion feature by disabling automatic deletions. Instead, we permit users to activate smart deletion on demand. For that, we used a tap and gesture hybrid approach, where users tap on the backspace key for regular backspace behaviors and perform a right to left gesture on the key to activate the smart-restorable backspace (Figure 3). If the inputted text contains multiple errors, the improved smart-restorable backspace targets the last committed one. To reach an error before that, the user has to smart-delete again, and so forth. This is to address the case that an error was intentionally left uncorrected, *e.g.*, an out-of-vocabulary word. In case of multiple errors within the same word, it deletes all letters following the earliest error, as this likely affects everything else in the word, too. In contrast, if the inputted text does not contain a spelling mistake, the smart-restorable backspace acts like the regular backspace, that is, deletes one character per gesture. This not only reduces the possibility of accidental and unexpected deletions but also the number of operations required to activate smart deletion. Now, only a single "left" stroke is necessary, while four backspaces were needed for smart deletion previously.

We also considered using long-press and double-tap on the key to activate smart deletion but decided against these, as they again interfere with the regular backspace behavior. In regular backspacing, long-press usually enables continuous character deletion, a feature also supported by the smart-restorable backspace. With double-tap, on the other hand, it is difficult to differentiate whether the intention was to activate smart deletion or to quickly delete two characters. Gestures do not suffer from this ambiguity. Besides, the fact that users are able to quickly learn and adapt to keyboards augmented with gestures [11], also motivated us to use this option.

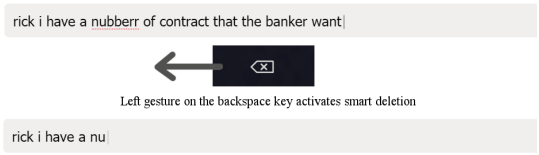

Figure 3. Activating smart deletion through a left gesture on the backspace key.

The new system also permits users to include a non-dictionary word into the dictionary by tapping on it in the text area. Once added to the dictionary, the system does not consider it as a misspelled word, eliminating the possibility of future accidental smart deletion. Similarly, users can delete the word from the dictionary by re-tapping on it.

Moreover, we improved the smart restoration feature by enabling it even for regular backspace operations. In the previous version of this feature, chunks of text that were deleted with repetitive keystrokes or long-press were not available for restoration. The improved version, in contrast, records all deleted text, increasing the restoration possibilities for the user. We also reduced the minimum key length from two to one character to display a restore suggestion earlier and with the minimum number of keystrokes possible. To facilitate access to smart restoration, we map activation to a right swipe on the backspace key. See Figure 4. We limit the smart deletion and smart restoration gestures, left and right swipes respectively, to the backspace key to account for the fact that some virtual keyboards map other functionalities to gestures [11], which makes our new method compatible with other keyboard systems.

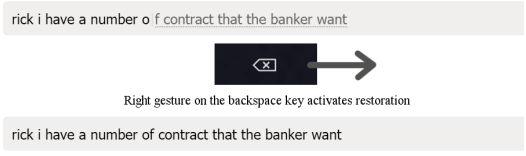

Figure 4. Visual feedback on a restorable text and smart restoration using a right gesture on the backspace key.

Finally, the improved smart restoration feature provides additional visual feedback for potentially restorable chunks.

To address the issue that users did not notice the prediction in the panel, the new version provides visual feedback for a restorable chunk directly in the text area in a grayed-out font (Figure 4). Then, the user can accept the suggestion by either performing a right gesture on the backspace key or tapping on the prediction in the prediction panel.

A USER STUDY

We conducted a user study to test the following hypotheses, which apply when users did not notice an error immediately after committing it:

H1: *The smart-restorable backspace performs better than the conventional backspace in terms of speed and efforts.*

H2: *The smart-restorable backspace is easy to learn and use, and users prefer it to the conventional backspace.*

Apparatus

We used multiple Microsoft Surface Pro 3 tablets, Intel Core i5 processor, 29.21×20.14×0.91 cm, and 798 grams, for the study, with Windows 8.1 at 2160×1440 resolution.

The Custom Application

We developed a custom Web application using HTML5 and JavaScript. We used the jQuery Mobile library to provide support for touch and gestures [22]. All text entry operations were processed on the client side to avoid delays. However, all user interactions were recorded in a MySQL/PHP database when the user pressed the enter key to submit a phrase. Figure 5 shows the device(s) and the application used in the study.

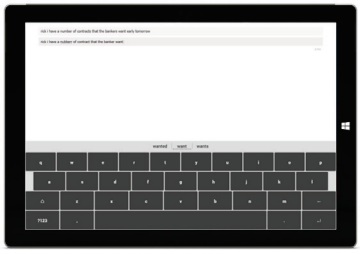

Figure 5. The device and the custom application used during the final user study.

Phrase Set

During the study, participants were asked to enter English phrases from the Enron set [28]. We used transcription tasks to increase the internal validity [29]. We decided to use the Enron corpus for two reasons. First, it is a collection of phrases written by actual mobile users on mobile devices, which increases the external validity of the study. Second, unlike most other sets, it contains phrases of various lengths, allowing us to test the new technique with relatively longer phrases. During the study we used a 775 phrase subset of the corpus, with 14-102 characters (5-15 words), and an average length of 41.45 (SD = 15.96). All phrases were presented in lowercase, without punctuation and special characters.

Design

We used a *between*-subjects design for the study. There were two groups (conditions): *conventional* backspace and the (improved) *smart-restorable* one. Participants were assigned to one of the two groups. Both groups had the same number of participants. We used a between-subjects design to collect as much data for each technique as possible, and to avoid any potential interference. There were two sessions, with roughly 24 hour gap in between. During the first session participants transcribed 80 random phrases. In the second session they transcribed 40 random phrases in two steps: first without and then with injected errors. Thus, there were 80 input attempts, as in the first session. In summary, the design was:

2 groups (*conventional* and *smart-restorable*) ×
10 participant per group ×
2 sessions, with about 24 hour gap in between ×
  Session 1: 80 phrases (excluding practice phrases) +
  Session 2: 40 phrases × (without + with injected error)
= 6,400 submissions, in total.

Error Injection

The purpose of the smart-restorable backspace is to facilitate the correction of "overlooked" errors, where the user did not notice or correct one or more mistake/s immediately after committing it. A prior investigation [6] found that such occurrences are infrequent, only 20-30% of a text entry episode, as also confirmed through post-hoc analysis of our pilot data. This makes it difficult to evaluate the new technique in a user study. Therefore, we artificially replicated error correction scenarios by injecting synthetic errors on the transcription. When users submit an error-free text by tapping on the enter button, the system injects an insertion, omission, substitution, or transposition error on the longest word. This is based on an observation that longer words are more prone to spelling errors. Besides, error injection on shorter words often results in other dictionary words, *e.g.*, "mam" to "man", making injected errors unnoticeable. If the maximum length occurs in multiple words in a phrase, the system randomly injected an error on one of the words. No errors were injected on text that already contained a spelling error.

We use these types of errors, as a prior study [21] showed that such errors represent about 90% of all those committed by experienced users. That study also resembled our use case the best, as we use comparatively large virtual keys (2.4×1.5 cm) and permit bimanual input. As in previous work [21], we simulate how such errors occur in the real-life, *i.e.*, 80% substitution errors replace a letter with a surrounding one from the same row. We also match the rates at which they occur, *i.e.*, 50% insertion, 26% substitution, 16% omission, and 8% transposition errors.

Participants

We recruited 20 paid participants from the university community. They were all experienced mobile touchscreen users. That is, they own and have been using a touchscreen-based mobile device on a daily basis, for at least three years. They were also experienced in mobile text entry, as all of them enter text on their mobile devices, *i.e.*, smartphone or tablet, almost every day. We randomly assigned them to each group, but made an effort to maintain roughly the same age range and gender ratio in the two groups.

Conventional: Ten participants were assigned to this group. Three were female. Their age ranged from 19 to 23, average 20.5 years (SD = 1.35). Seven were right-handed, one left-handed, and one ambidextrous.

Smart-Restorable: Ten participants were assigned to this group. Two were female and age ranged from 19 to 28, average 22 years (SD = 3.35). All were right-handed.

Procedure

At the start of the first session, we demonstrated the technique corresponding to their group to them. We made sure they understood and knew how to use all features. Then, we asked them to practice with the system, where they had to enter at least three phrases using "their" technique. However, they were allowed to extend this practice period on request. The intention was to make sure they felt confident using the technique. The main study started after that, where we asked the participants to transcribe eighty English phrases from our phrase set, which were shown in random order on the display. We instructed participants to read and understand the phrases in advance, then to enter them as quickly and accurately as possible. When finished, they had to press the enter key to see the next phrase. Error correction was *forced* during the study and participants had to correct all mistakes before submitting it, otherwise the system made a "ding" noise to notify them of any existing error. While, the system supported direct cursor control, we disabled this in the study to eliminate a potential confound. Hence, participants were asked to exclusively use backspace for error correction. We permitted them to position the tablet on the table in any comfortable landscape position (Figure 6). We also allowed participants to use any posture they felt conformable with for text entry. All of them ended up using both hands.

Figure 6. A participant inputting text using the custom application during the final user study.

The second session started the next day and followed the same procedure as the first one. In addition, error injection was added to each submission attempt. That is, the system injected an insertion, substitution, omission, or transposition error on the longest word of the phrase upon submission. However, users were only informed that errors would occur on a random word. Participants had to correct that mistake and resubmit to see the next phrase. We decided to inject errors on each submitted phrase to reduce the effort for reading and understanding a new phrase. Participants were informed of this behavior ahead of time to avoid any surprise effects. We explained that this was designed to simulate a situation where they overlooked a mistake. Participants were able to test this in a practice period, where they were asked to enter at least three phrases. They could extend this practice period upon request.

Upon completion of the study, participants were asked to fill out a short questionnaire where they could rate the technique they used on a seven-point Likert scale and also to comment on the techniques and study procedure.

Metrics

In addition to calculating words per minute (wpm), operations per character, and backspace rate as in the pilot study, here we also calculated the following metrics.

- **Visual scan time (milliseconds):** denotes the average time it takes to verify an inputted phrase. This was calculated by measuring the interval between the last character entry and the enter key press [5].
- **Backspace time (milliseconds):** denotes the average time it takes the user to perform a backspace. This metric was calculated by measuring the interval between the last character entry and a backspace. Apart from the decision, preparation, and action time for the backspace, this also includes the visual scan for the last character. We did not separate these human processes as cognitive modeling is outside the scope of our work. This metric excludes long-press backspace incidents.
- **Post-backspace time (milliseconds):** signifies the average time it takes to input a character after performing a backspace. This was calculated by measuring the interval between a backspace and the next character. This metric is the compound of the verification time for the backspace and the decision, preparation, and action time for the next character.

RESULTS

We analyzed the data with mixed ANOVA for the effect of technique (between subjects) and session (within). In the figures, error bars represent ±1 standard error.

Entry Speed

An ANOVA revealed that there was no significant effect of technique on text entry speed ($F_{1,18}$ = 0.1, ns) across all sessions. Average entry speeds for conventional and smart-restorable were 27.75 (SE = 0.28) and 28.36 wpm (SE = 0.27), respectively. There was a significant effect of session ($F_{1,18} = 27.21$, $p < .0001$, $\eta^2 = .03$), but no technique × session interaction ($F_{1,18}$ = 0.2, ns). Figure 7 shows average entry speed for both techniques during the two sessions.

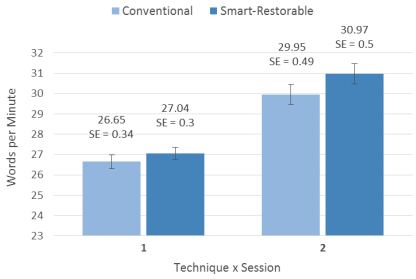

Figure 7. Average entry speed for both techniques by session.

An ANOVA identified a significant effect of technique on entry speed for phrases with injected errors ($F_{1,18} = 4.49$, $p < .05$, $\eta^2 = .03$). Average entry speeds for conventional and smart-restorable were 18.75 (SE = 0.46) and 21.83 wpm (SE = 0.4), respectively. See Figure 8.

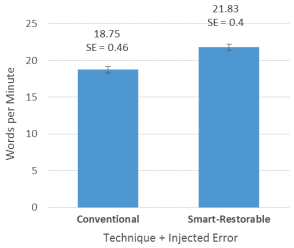

Figure 8. Average entry speed for both techniques in the injected error block.

Operations per Character

There was no significant effect of technique on operations per character ($F_{1,18} = 0.11$, ns). On average conventional and smart-restorable required 1.2 (SE = 0.01) and 1.2 (SE = 0.01) operations per character, respectively. There was also no significant effect of session ($F_{1,18} = 1.92$, $p > .05$) and the interaction was not significant, either. Figure 9 displays average operations per character for both techniques during the two sessions.

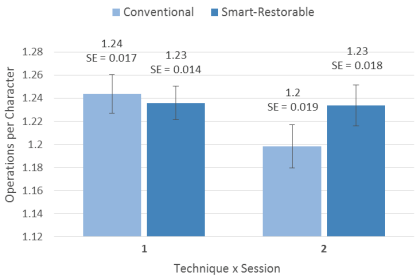

Figure 9. Average operations per character for both techniques by session.

An ANOVA identified a significant effect of technique on operations per character for phrases with injected errors ($F_{1,18} = 72.14$, $p < .00001$, $\eta^2 = .16$). Average operations per character with injected error for conventional and smart-restorable were 2.1 (SE = 0.04) and 1.52 (SE = 0.02), respectively. See Figure 10.

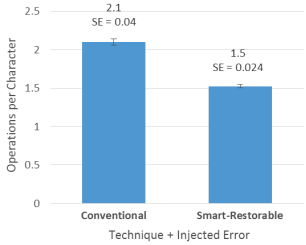

Figure 10. Average operations per character for both techniques in the injected error block.

Backspace Rate

An ANOVA did not find a significant effect of technique on backspace rate ($F_{1,18} = 1.79$, $p > .05$). On average 7.71 (SE = 0.24) and 8.96% (SE = 0.28) of all keystrokes were backspaces for conventional and smart-restorable, respectively. There was also no significant effect of session ($F_{1,18} = 2.68$, $p > .05$) or technique × session ($F_{1,18} = 0.19$, ns). Figure 11 shows the average backspace rate for both methods in the two sessions.

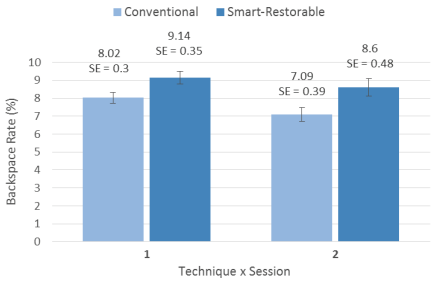

Figure 11. Average backspace rate for both techniques by session.

There was a significant effect of technique on backspace rate for phrases with injected errors ($F_{1,18} = 142.99$, $p < .00001$, $\eta^2 = .22$). Average backspace use was 23.6 (SE = 0.56) and 12.3% (SE = 0.5) for conventional and smart-restorable, respectively. Figure 12 illustrates this.

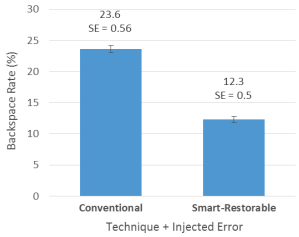

Figure 12. Average backspace rate for both techniques in the injected error block.

A deeper investigation revealed that in the conventional condition about 6.82% and 1% of all inputted characters were deleted using the regular and tap-hold methods, respectively. Interestingly, smart-restorable had similar results, where about 7.6%, 1.3%, and 0.1% of all entered characters were deleted using regular, tap-hold, and the smart gesture, respectively. Also, about 1.32% of all characters were restored using the smart restore feature. As expected, these rates increased with injected errors.

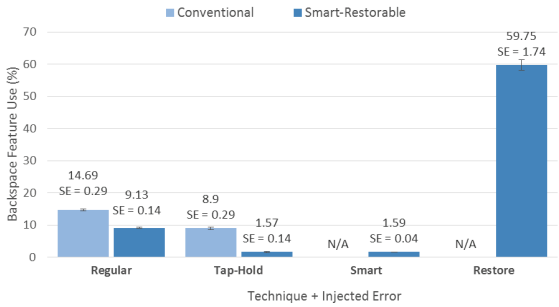

Figure 13. Average backspace feature distribution for both techniques in the injected error block.

A Mann-Whitney U test revealed that the rates in which regular and tap-hold methods were used, were significantly

different for different techniques for phrases with injected errors. Both regular ($U = 36413.0$ $Z = -13.339$, $p < .0001$) and tap-hold ($U = 53170.0$ $Z = -10.982$, $p < .0001$) rates reduced significantly for smart-restorable. Yet, about 1.59% of all inputted characters were deleted using the smart gesture and 60% were restored using the smart restore feature (Figure 13).

Visual Scan Time

An ANOVA did not find a significant effect of technique on visual scan time ($F_{1,18} = 0.83$, ns). The average visual scan time for conventional and smart-restorable were 840.22 (SE = 17.3) and 743.22 (SE = 16.027) ms, respectively. However, there was also a significant effect of session ($F_{1,18} = 9.41$, $p < 0.01$, $\eta^2 = .01$) but not on technique × session ($F_{1,18} = 1.80$, $p > .05$). Figure 11 shows the average backspace rate for both techniques during the two sessions.

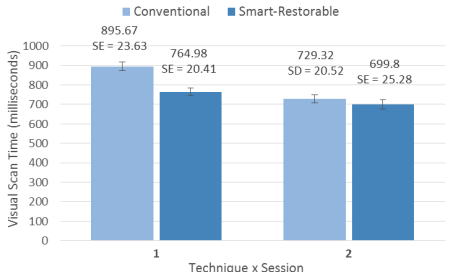

Figure 14. Average visual scan time for both techniques by session.

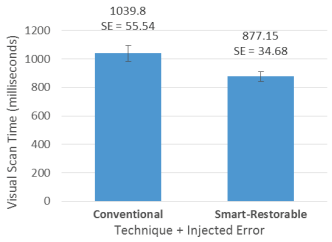

Figure 15. Average visual scan time for both techniques in the injected error block.

There was no significant effect of technique on visual scan time with injected errors ($F_{1,18} = 2.51$, $p > .05$). Average visual scan times with injected error were 1039.8 (SE = 55.5) and 877.15 (SE = 34.7) ms for conventional and smart-restorable, respectively. Figure 15 illustrates this.

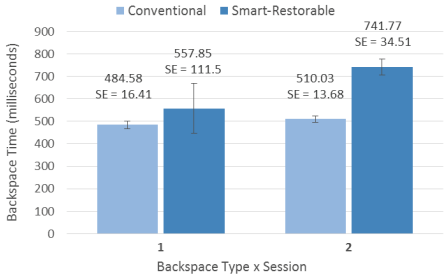

Figure 16. Average backspace time for both backspace types by session.

Backspace Time

An ANOVA failed to find a significant difference between backspace type in terms of operation time ($F_{1,18} = 1.45$, $p > .05$). On average performing the conventional and the smart-restorable backspace took 495.6 (SE = 11.04) and 717.08 (SE = 33.5) ms, respectively. An ANOVA also failed to find a significant effect of session ($F_{1,18} = 1.19$, $p > .05$) or backspace type × session ($F_{1,18} = 0.68$, ns). Figure 16 illustrates average backspace times for both techniques during the two sessions.

Post-Backspace Time

An ANOVA identified a significant effect of backspace type on post-backspace time ($F_{1,18} = 9.10$, $p < .01$, $\eta^2 = .002$). On average operations following the conventional and the smart-restorable required 580.3 (SE = 8.96) and 1171.5 (SE = 55.4) ms, respectively. However, there was no significant effect of session ($F_{1,18} = 3.83$, $p = .06$) or technique × session ($F_{1,18} = 1.9$, $p > .05$). Figure 17 shows the average post-backspace time for the two backspace methods during the two sessions.

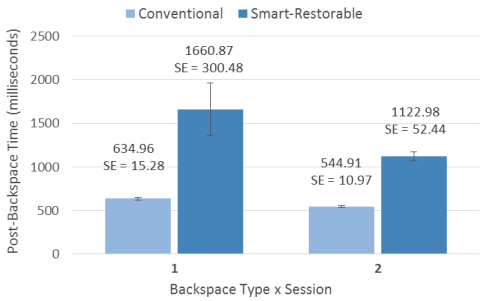

Figure 17. Average post-backspace operation time for both backspace types by session.

Prediction Behaviors

An ANOVA did not find a significant effect of technique on accepted prediction ($F_{1,18} = 0.83$, ns) or autocorrection ($F_{1,18} = 0.90$, ns) rates. On average 8.97 and 8.04% of all predicted words were accepted in conventional and smart-restorable, respectively. Similarly, about 5.5 and 6.9% of all supposedly misspelled words were autocorrected in conventional and smart-restorable, respectively (Figure 18). However, in both techniques, about 1.7% (SE = 0.10) words were incorrectly autocorrected. These rates were similar with injected errors.

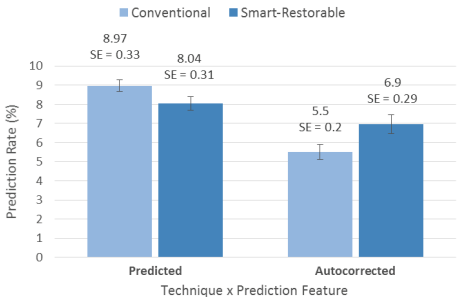

Figure 18. Average predictive feature distribution for both techniques.

User Feedback

We converted all seven-point scales to three-point scales using linear transformation to calculate ratios (%), which is common practice in statistics [13].

Conventional Backspace: Most participants, 90%, agreed that error correction is difficult with the conventional backspace,

while the remaining 10% disagreed. Besides, 50% believed that conventional backspace compromises their entry speed, while 30% did not think that it has any impact on their performance. The remaining 20% participants were neutral regarding this. Yet, none of them were fully satisfied with the current backspace technique, and 90% wanted a better correction method, while 10% were impartial.

Smart and Restorable Backspace: Participants were mostly positive about the smart-restorable backspace. Almost all of them (90%) found it easy to learn and use, while the remaining 10% were neutral. Nonetheless, all of them (100%) thought that it made error correction faster and easier than the methods they use on their mobile devices. 90% thought that it increased their overall entry speed, while the remaining 10% were impartial. Overall, 90% participants liked smart-restorable more than the techniques they use, and wanted to keep using it. The remaining 10% were neutral.

DISCUSSION

The results reveal that the two examined techniques were not significantly different in terms of speed and operations per character. We expected this, as error correction, which the new technique addresses, occurs relatively infrequently in text entry. There was a significant effect of session on speed, suggesting that with practice entry speed improves substantially for both techniques. Also, the smart-restorable backspace yielded on average 2.2% better entry speed than the conventional technique, with a comparable number of operations per character. This is an obvious improvement over our pilot, where the new technique had lower entry speed. This indicates that the redesign improved the technique's performance. Also, as predicted, both speed and operations per character improved significantly with injected errors. That is, in situations where users attempted to correct an "overlooked" error, the new technique improves entry speed by 16.5% and reduces operation per character by 27.6%.

Backspace rates were not significantly different for the investigated techniques. However, with injected errors, the new technique yielded a significantly (48%) lower backspace rate than the conventional one. A deeper analysis revealed that about 60% of all backspace gestures were used to restore text. This suggests that with the smart-restorable backspace, users used the backspace gestures more to restore text than to delete it. This is most probably due to the fact that the smart restore feature splits deleted text based on the *CP*s to avoid restoration of incorrect texts, which provides the user with restoration options for each error correction. This suggests that users benefited the most from this feature. Thus we recommend its inclusion in all predictive keyboards. Interestingly, there was no significant effect of technique on visual scan time, even with injected errors, which means users did not take additional time to verify an input using the new technique, even when the input contained errors.

Apart from these metrics, we also analyzed the performance of different backspace actions. While there was no significant difference, performing a smart-backspace required on average *only* about 222 milliseconds more than the regular backspace. This is surprising, as prior studies found gesture-based input to be significantly slower than tap-based input [4]. We speculate that the reason is that we instructed our participants to perform the gestures "on the backspace key", instead of "on the keyboard", which reduced the average length of a gesture, resulting in faster operation time. The fact that we only recruited experienced touchscreen users may have contributed to this as well, as studies showed that users tend to use shorter gestures with experience [9]. However, there was a significant effect of backspace type on post-backspace operation time. On average users took 591 milliseconds more time to perform an operation following a smart-backspace. The most probable reason is that users needed extra time to prepare for the next step, after deleting a chunk of text. Yet, this time reduced by 32% in the second session, suggesting that post-backspace time for the smart-restorable backspace may reduce further with practice. In the second session and for the backspace and post-backspace time combined, the smart-restorable backspace required about 810 milliseconds more time than the conventional one, which is an obvious improvement from the pilot's 2.1 second overhead. Thus, the new technique improves the overall text entry performance, by significantly reducing correction efforts for "overlooked" mistakes, which supports acceptance of H_1.

Almost all participants (90%) in the conventional group were unhappy with the technique and felt the necessity for a better approach. 50% of them also felt that it reduces their text entry performance. Participants in the smart-restorable group were mostly positive about the examined technique. All of them (100%) rated it higher than the method they use on their tablets and most of them (90%) wanted to keep using it. Most of them (90%) also found it easy to learn and use, and improved their overall text entry performance. These results support the acceptance of H_2.

CONCLUSION AND FUTURE WORK

We presented a new smart-restorable backspace technique to facilitate the correction of "overlooked" errors. We compared the technique with the conventional backspace in a user study. Results showed that the new technique improves the overall text entry performance, in terms of speed and operation per character, by significantly reducing error correction efforts for "overlooked" mistakes. Results also revealed that most users liked the new approach better than the one they use on their devices, and found it easy to learn and to use. Most of them also felt that it improves their overall text entry performance and wanted to keep using it.

In the future, we intend to explore the possibility of a word-based deletion feature to address cases where the system fails to identify an error, such as an incorrectly autocorrected word.

ACKNOWLEDGMENTS

We thank the attendees of the CHI '15 Workshop on Text Entry on the Edge [19] for their feedback on the study design. This work has been supported in part by the Canada Research Chairs program, NSERC, and NCE.

REFERENCES

1. Apple Insider. First Look: iOS 9's New Cursor-Controlling Gesture Keyboard on iPad. 2015. Retrieved August 22, 2015 from http://appleinsider.com/articles/15/06/11/first-look-ios-9s-new-cursor-controlling-gesture-keyboard-on-ipad
2. Apple Insider. How to use iOS 9's keyboard as a trackpad with 3D Touch on iPhone 6s. 2015. Retrieved December 30, 2015 from http://appleinsider.com/articles/15/10/11/how-to-use-ios-9s-keyboard-as-a-trackpad-with-3d-touch-on-iphone-6s-
3. Ahmed Sabbir Arif, Mauricio H. Lopez, Wolfgang Stuerzlinger. 2010. Two new mobile touchscreen text entry techniques. *Poster at the 36th Graphics Interface Conference* (GI '10). CEUR-WS.org/Vol-588.
4. Ahmed Sabbir Arif, Michel Pahud, Ken Hinckley, and Bill Buxton. 2014. Experimental study of stroke shortcuts for a touchscreen keyboard with gesture-redundant keys removed. In *Proceedings of Graphics Interface 2014* (GI '14). Canadian Information Processing Society, Toronto, Ont., Canada, Canada, 43-50.
5. Ahmed Sabbir Arif and Wolfgang Stuerzlinger. 2009. Analysis of text entry performance metrics. In *Proceedings of IEEE Toronto International Conference on Science and Technology for Humanity* (TIC-STH '09). IEEE, Washington, DC, USA, 100-105.
6. Ahmed Sabbir Arif and Wolfgang Stuerzlinger. 2010. Predicting the cost of error correction in character-based text entry technologies. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '10). ACM, New York, NY, USA, 5-14.
7. Ahmed Sabbir Arif and Wolfgang Stuerzlinger. 2013. Pseudo-pressure detection and its use in predictive text entry on touchscreens. In *Proceedings of the 25th Australian Computer-Human Interaction Conference: Augmentation, Application, Innovation, Collaboration* (OzCHI '13), Haifeng Shen, Ross Smith, Jeni Paay, Paul Calder, and Theodor Wyeld (Eds.). ACM, New York, NY, USA, 383-392.
8. Ahmed Sabbir Arif, Wolfgang Stuerzlinger, Ali Mazalek, Sunjun Kim, and Geehyuk Lee. 2015. A smart-restorable backspace to facilitate text entry error correction. In *CHI 2015 Workshop on Text Entry on the Edge* (April 18, 2015). Seoul, South Korea.
9. Ahmed Sabbir Arif, Wolfgang Stuerzlinger, Euclides Jose de Mendonca Filho, and Alec Gordynski. 2014. Error behaviours in an unreliable in-air gesture recognizer. In *Proceedings of the Extended Abstracts of the 32nd Annual ACM Conference on Human Factors in Computing Systems* (CHI EA '14). ACM, New York, NY, USA, 1603-1608.
10. Geoffroy Aulagner, Romain François, Benoît Martin, Dominique Michel, and Mathieu Raynal. 2010. Floodkey: increasing software keyboard keys by reducing needless ones without occultation. In *Proceedings of the 10th WSEAS International Conference on Applied Computer Science* (ACS' 10), Hamido Fujita and Jun Sasaki (Eds.). World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA, 412-417.
11. Matt Bolton. 2014. How to Improve iOS Autocorrect. Mac Life. Retrieved July 21, 2015 from http://www.maclife.com/article/howtos/how-improve-autocorrect
12. James Clawson, Kent Lyons, Alex Rudnick, Robert A. Iannucci, Jr., and Thad Starner. 2008. Automatic whiteout++: correcting mini-QWERTY typing errors using keypress timing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '08). ACM, New York, NY, USA, 573-582.
13. John G. Dawes. 2008. Do data characteristics change according to the number of scale points used? An experiment using 5 point, 7 point and 10 point scales. *International Journal of Market Research 51*, 1.
14. Paul M. Fitts. 1954. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology* 47, 6: 262-269.
15. Fleksy Keyboard. 2015. Retrieved August 12, 2015 from https://fleksy.com
16. Vittorio Fuccella, Poika Isokoski, and Benoit Martin. 2013. Gestures and widgets: performance in text editing on multi-touch capable mobile devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '13). ACM, New York, NY, USA, 2785-2794.
17. Gartner, Inc. Forecast: PCs, Ultramobiles, and Mobile Phones, Worldwide, 2011-2018, 2Q14 Update. 2015. Retrieved July 21, 2015 from https://www.gartner.com/doc/3077322/forecast-pcs-ultramobiles-mobile-phones
18. Joshua Goodman, Gina Venolia, Keith Steury, and Chauncey Parker. 2002. Language modeling for soft keyboards. In *Proceedings of the 7th International Conference on Intelligent User Interfaces* (IUI '02). ACM, New York, NY, USA, 194-195.
19. James Clawson, Ahmed Sabbir Arif, Stephen Brewster, Mark Dunlop, Per Ola Kristensson, and Antti Oulasvirta. 2015. Text Entry on the Edge. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems* (CHI EA '15). ACM, New York, NY, USA, 2381-2384.

20. Google Code. The Diff Match and Patch Libraries. 2012. Retrieved August 10, 2015 from https://code.google.com/p/google-diff-match-patch
21. Jonathan T. Grudin. 1983. Error patterns in novice and skilled transcription typing. In *Cognitive Aspects of Skilled Typewriting*, W. E. Cooper (Ed.). Springer-Verlag, 121-143.
22. jQuery Mobile. A Touch-Optimized Web Framework. 2015. Retrieved March 12, 2015 from https://jquerymobile.com
23. Sunjun Kim and Geehyuk Lee. 2012. Restorable backspace. In *Adjunct proceedings of the 25th annual ACM symposium on User interface software and technology* (UIST Adjunct Proceedings '12). ACM, New York, NY, USA, 73-74.
24. KeuKey Keyboard. 2014. Retrieved July 21, 2015 from http://www.keukey.com
25. Vladimir I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet Physics Doklady 10*, 8: 707-710.
26. I. Scott MacKenzie and R. William Soukoreff. 2003. Phrase sets for evaluating text entry techniques. In *CHI '03 Extended Abstracts on Human Factors in Computing Systems* (CHI EA '03). ACM, New York, NY, USA, 754-755.
27. Dominik Schmidt, Florian Block, and Hans Gellersen. 2009. A Comparison of Direct and Indirect Multi-touch Input for Large Surfaces. In *Proceedings of the 12th IFIP TC 13 International Conference on Human-Computer Interaction: Part I* (INTERACT '09), Tom Gross, Jan Gulliksen, Paula Kotzé, Lars Oestreicher, Philippe Palanque, Raquel Oliveira Prates, and Marco Winckler (Eds.). Springer-Verlag, Berlin, Heidelberg, 582.
28. Keith Vertanen and Per Ola Kristensson. 2011. A versatile dataset for text entry evaluations based on genuine mobile emails. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services* (MobileHCI '11). ACM, New York, NY, USA, 295-298.
29. Keith Vertanen and Per Ola Kristensson. 2014. Complementing text entry evaluations with a composition task. *ACM Transactions on Computer-Human Interaction* 21, 2, Article 8 (February 2014), 33 pages.
30. Wiktionary. The Most Common Words in Project Gutenberg: Frequency lists/PG/2006/04/1-10000. 2006. Retrieved July 21, 2015 from https://en.wiktionary.org/wiki/Wiktionary:Frequency_lists/PG/2006/04/1-10000