Text Text Revolution: A Game that Improves Text Entry on Mobile Touchscreen Keyboards

Dmitry Rudchenko, Tim Paek[1], Eric Badger

[1] Microsoft Research and Microsoft Corporation,
One Microsoft Way, Redmond, WA 98052 USA
{dmrudche, timpaek, ebadger}@microsoft.com

Abstract. Mobile devices often utilize touchscreen keyboards for text input. However, due to the lack of tactile feedback and generally small key sizes, users often produce typing errors. Key-target resizing, which dynamically adjusts the underlying target areas of the keys based on their probabilities, can significantly reduce errors, but requires training data in the form of touch points for intended keys. In this paper, we introduce Text Text Revolution (TTR), a game that helps users improve their typing experience on mobile touchscreen keyboards in three ways: first, by providing targeting practice, second, by highlighting areas for improvement, and third, by generating ideal training data for key-target resizing as a side effect of playing the game. In a user study, participants who played 20 rounds of TTR not only improved in accuracy over time, but also generated useful data for key-target resizing. To demonstrate usefulness, we trained key-target resizing on touch points collected from the first 10 rounds, and simulated how participants would have performed had personalized key-target resizing been used in the second 10 rounds. Key-target resizing reduced errors by 21.4%.

Keywords: Game, key-target resizing, text entry, touchscreen keyboard

1 Introduction

Mobile devices with capacitive or resistive touch sensors often utilize an on-screen, virtual keyboard (see [10] for a survey), or *touchscreen keyboard*, for text input. Without the need for dedicated hardware, touchscreen keyboards facilitate larger displays for videos, web pages, email, etc. [11]. As software, touchscreen keyboards can easily accommodate different languages, screen orientation, and key layouts. On the other hand, touchscreen keyboards lack the tactile affordances of a physical keyboard, which have been shown to be critical for touch typing [15]. Due to the lack of tactile feedback and generally small key sizes, users often produce typing errors. To reduce noisy input, researchers have developed algorithms for dynamically adjusting the underlying target areas of keys based on probabilities, a technique called *key-target resizing*. As shown in previous research [8,9], key-target resizing can significantly reduce typing errors, but requires labeled training data in the form of touch points for intended keys. In this paper, we introduce *Text Text Revolution* (TTR), a game that helps users improve their typing experience on mobile touchscreen keyboards in three ways: first, by providing targeting practice, second, by

highlighting areas for improvement, and third, by generating ideal training data for key-target resizing as a side effect of playing the game. This paper is organized as follows. In Section 2, we describe how we designed the game to address specific text entry goals. In Section 3, we discuss how TTR relates to prior research on leveraging human computation. Finally, in Section 4, we evaluate how well TTR accomplishes its text entry goals through a user study and simulation experiment.

2 Game Design and Text Entry Goals

Because text entry on mobile touchscreen keyboards can be quite challenging, a small market has opened up for mobile typing apps, such as *SpeedType* [17] for the iPhone. In most typing apps, users are prompted with text they are required to type. When users mistype a character, they are typically notified through auditory and visual feedback, such as beeps and squiggly underlines. In some cases, the feedback can be more implied. For example, in *Turbo Type* [18], a race car representing the user slows down with typing errors. When users are finished typing, they typically receive information about their text entry speed and accuracy, but not information highlighting areas for improvement. Instead, users are typically encouraged to practice and to beat their personal best scores.

Inspired by this market for typing apps, we endeavored to design a text entry game with three goals in mind. First, like all typing applications, we sought to provide users with lots of practice targeting the keys of a mobile touchscreen keyboard. After all, the rendered keys can be quite small. For example, on the iPhone, most adult fingers easily cover two to three keys. Second, unlike most typing apps, we sought to provide users with concrete means to improve their typing by visually highlighting keys they tend to mistype. Third, we sought to obtain "ideal" training data for key-target resizing. As formalized in [8], key-target resizing employs a probabilistic approach to decoding noisy touch input. It combines probabilities from both a *language model* for predicting the likelihood of a next character given previous characters and a *touch model* for predicting the likelihood of observing a touch point (e.g., pixel coordinate or ellipsoid) given the intention to hit various keys.

Unfortunately, most of the time, it is not possible to know without inference what keys users are intending to hit, *unless*, of course, we instruct them to hit those keys. This is the hidden treasure of typing apps. By giving users text they are supposed to type, these apps are acquiring a wealth of labeled touch point data which can be leveraged immediately to learn a touch model for every key. For example, if, whenever users are instructed to type 'g', they correctly hit 'g' with some frequency and 'v' with some other frequency, we are essentially learning a probability distribution over likely touch points (mapped to keys) given the intention to type 'g'. Furthermore, if we can train touch models on the fly using data from the game, then the more users play the game, the more robust key-target resizing will be – which translates into reduced typing errors on the soft keyboard (we empirically demonstrate this in Section 4). Besides real-time adaptation, we could also aggregate user touch data in the cloud (i.e., on web servers) and leverage collaborative filtering to learn touch models for similar patterns of touch points for keys. The cloud could then push

Figure 1. Screenshots of Text Text Revolution: (a) Sample text users are instructed to type; (b) Letters exploding when users touch an expected or adjacent key; (c) End screen highlighting areas for improvement.

these touch models down to the mobile device. In short, by having a game that allows us to collect touch points for intended keys, we can explore opportunities for real-time adaptation and collaborative filtering.

In terms of implementation, we developed TTR using the XNA game development language and the Windows Phone 7 SDK [21]. The game runs on any Windows Phone 7 device and is available for free in the Windows Phone Marketplace.

2.1 Game Play

With the above goals in mind, the game play of TTR proceeds as follows. As shown in Figure 1(a), users are prompted with words they are instructed to type. The words are randomly selected from a corpus of 10,000 words. Following [15], the corpus was generated by minimizing the relative entropy of character bigrams in the corpus with respect to a larger source – in our case, over 1 million email messages and transcribed voicemail messages. This allows us to provide users with consecutive characters that are representative of consecutive characters in email, a common mobile task. When users touch the keys we expect, the letters explode forward and fade out, as depicted in Figure 1(b). When users touch keys that are immediately adjacent to the keys we expect, we still explode the letters (e.g., ‘q’, ‘w’, ‘s’, ‘z’ for ‘a’). We did this for two reasons. First, we noticed that in typing apps where users are not permitted to move forward unless they hit the correct key, users tend to change their normal typing behavior; in particular, they tended to slow down and more carefully hunt-and-peck

each key. Because we are interested in collecting "natural" touch points, we decided to relax the direct hit requirement. Second, we wanted to simulate an "ideal" typing experience. If key-target resizing could accurately infer intended keys despite the user hitting adjacent keys, then the experience of typing with unerring key-target resizing would be identical to what the game simulates. In essence, the game allows users to experience the kind of typing experience we hope to enable by collecting their labeled touch points and training key-target resizing on their data. If users hit a key that is neither the expected key nor an adjacent key, the text turns red and a beep sound is played. This immediately alerts users to their errors.

As shown in Figure 1(a) and (b), on each side of the presented text are two bars. The left bar indicates a running estimate of typing speed expressed in *words-per-minute* (WPM), and the right bar indicates a running estimate of typing accuracy. *Accuracy* is simply computed as the number of touch points that directly hit the intended key divided by the number of keystrokes. Users complete a word when they finish entering all of the word's letters. To move to subsequent words, users touch the space bar or any of the adjacent keys above the space bar. Because users sometimes miss the space bar, we can also learn a probability distribution for the space bar.

At the end of each game, we present users with a map of all of their accumulated touch points on the keyboard layout, as shown in Figure 1(c). The touch points are colored for each target key, which allows users to easily see when their touch points might be encroaching into unintended keys. For example, in Figure 1(c), most of the colored touch points for the letter 'l' are inside the boundaries of the 'l' key. However, some touch points bleed into the boundaries of the 'k'. As highlighted by the orange squares, the user in this case can immediately see areas for improvement; namely, avoid mistyping 'k' for 'l', and 'b' and 'n' for the space bar. When the user presses the "Submit Score" button, the score is sent to the cloud and the user receives a screen displaying a scoreboard containing the user's best WPM and accuracy scores as well as the top scores from any user. However, even before users have the opportunity to submit their scores it is important to note that we send all of their data to the cloud for training key-target resizing. This is done with the user's permission via a privacy dialog box that is presented the first time the user launches the game.

3 Related Research

The problem of attaining touch point data to train key-target resizing can be viewed as part of the larger challenge of leveraging human computation for useful purposes. One method that has been gaining momentum in the research community is the online platform *Amazon Mechanical Turk* [1], which allows developers to incorporate paid human intelligence via crowdsourcing into their applications. Indeed, due to the generally low cost of data, researchers have begun to exploit Mechanical Turk for natural language processing tasks related to text entry, such as transcribing native [14] and non-native speech [6]. Another method is to exchange human computation for entertainment in the context of a computer game. According to the Entertainment Software Association, 67% of American households play computer or videogames [5]. This has prompted researchers, most notably Luis Von Ahn and colleagues, to

design clever *games with a purpose* (GWAP), in which human players perform tasks which computers cannot automate easily as part of the game (see [19] for a survey). For example, in *The ESP Game* [20], players generate meaningful, accurate labels for images on the Web as they try to guess what their game partners are thinking. In essence, they are producing labeled training data for an object recognition system. Amazingly, as of July 2008, 200,000 players contributed more than 50 million labels.

While TTR falls under the rubric of a GWAP, it is also very akin to training wizards that not only teach users how to perform a task, but gather adaptation data along the way. For example, in the Windows 7 Speech Recognition Tutorial [22], users not only learn and practice voice commands for accessing features of the Windows 7 operating system by voice, but they also contribute example pronunciations for various phonemes, or segmental units of sound, that make up a language. In fact, the Tutorial has a section where users are presented with text they are instructed to read aloud. When they pronounce each word correctly, they are allowed to proceed. The acoustic data generated by the Tutorial is then used to adapt speech engine parameters. The entire setup of the Tutorial is more or less the same as what we use for TTR, except that we use a game instead of a wizard to entice users.

4 Evaluation

In order to evaluate TTR, we conducted a user study in which we recruited 6 participants, half of whom were female, to play the game. The participants played 20 rounds of TTR on a 3.5 inch WVGA, capacitive touchscreen device. Each round consisted of 250 characters, or approximately 50 words. Participants were told to use whatever posture for inputting text on the keyboard felt comfortable (e.g., two thumbs, one thumb, etc.) so long as they consistently used that posture for all subsequent rounds. Any time participants needed a break, they could pause after completing a word during a round of TTR, or they could relax before the next round. Participants were all employees of Microsoft and were compensated for their time.

Before assessing how well TTR achieves its text entry goals, one important question to ask is whether or not the game is engaging. Since the game was released, it has been downloaded by over 25,000 unique users. 134 provided ratings [2], with an average score of 4.5/5 stars. The vast majority of raters posted positive comments about both its gameplay and usefulness, such as "*This is so much FUN! I can't stop playing, I think I'm addicted*", "*The leaderboard feature makes it a lot of fun*", and "*Very additive game. Very useful as well. My mobile typing is ten times faster now*."

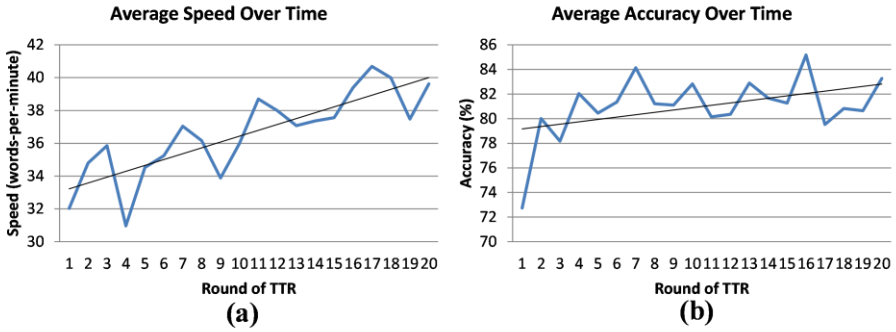

Figure 2. Targeting practice improving (a) speed and (b) accuracy over time (multiple rounds of TTR). Trend line has been added in.

With respect to text entry goals, the first goal of TTR was to provide users with targeting practice so they could improve over time by sheer repetition. Not surprisingly, participants improved in both speed and accuracy over the 20 rounds, as shown by the trend lines in Figure 2(a) and (b). The second goal of TTR was to provide users with areas for improvement. TTR accomplishes this by visually displaying where users tended to mistype keys on a map of touch points overlaid on the keyboard. According to a post-hoc questionnaire, 4/6 participants found the touch point map to be useful. Of the 2 participants who did not find it useful, they claimed to have not even noticed the map. We are considering methods to make suggestions inherent in the touch point map more salient (e.g., written suggestions).

4.1 Simulation Experiment

Finally, in order to assess the third goal of TTR, generating useful training data for key-target resizing, we trained key-target resizing on touch points collected from the first 10 rounds of the participants' data, and simulated how participants would have performed had key-target resizing been used in the second 10 rounds. We are able to simulate performance by taking the touch points collected from TTR and seeing if key-target resizing would have changed the key assigned to each touch point. For example, suppose the user was attempting to type an 'a' and touched an (x,y) coordinate on the keyboard "normally" corresponding to an 's', where "normally" means using a key mapping that is based on fixed key boundaries which are equidistant in both the vertical and horizontal axis from neighboring keys. Recall that key-target resizing dynamically adjusts the key boundaries based on probabilities. It does this by simply taking an (x,y) coordinate and returning a key assignment. As long as key-target resizing is constrained to have convex target regions [9], it will assign the same key to any two touch points contained within the target region. We can now investigate whether key-target resizing would have assigned an (x,y) coordinate that was incorrectly assigned to an 's' to an 'a' instead. If key-target resizing would have assigned the (x,y) coordinate to an 'a', then the number of direct hits, and hence accuracy, would have increased. Likewise, suppose the user had touched an (x,y) coordinate normally corresponding to a direct hit. If key-target

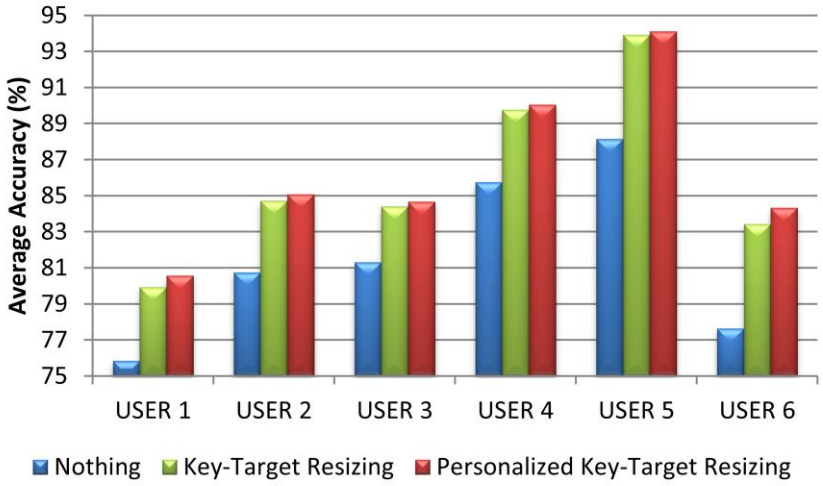

Figure 3. Simulation results showing how well users would have done on the 11-20th rounds of TTR in accuracy had key-target resizing been on (green) and had key-target resizing been personalized to each user's data (red).

resizing would have assigned a different key to that (x,y) coordinate, then the number of direct hits, and hence accuracy, would have decreased.

Figure 3 displays the simulation results for the 6 participants. Overall, no participant would have achieved an average accuracy higher than 89% in the last 10 rounds. However, if we had applied key-target resizing, using parameters trained on general data, the average accuracy would have jumped up from 78.2% to 82.4% ($t(5)=10.0$, $p<.001$), a relative error reduction of 18.9%. In all cases, key-target resizing would have been beneficial and for some participants (viz., user 5 and 6) it would have resulted in a dramatic increase in accuracy, pushing user 5 into the 90% range. For "personalized" key-target resizing, we trained the algorithms on each user's touch points from the first 10 rounds of TTR and only that user's data. As evident in Figure 3, personalized key-target resizing consistently improves accuracy across all participants. The average accuracy of personalized key-target resizing is 82.9%, which constitutes a relative error reduction of 21.4% over key-target resizing ($t(5)=-9.8$, $p<.001$). Note that the personalized data we used for training was smaller than the general data for key-target resizing. As such, it is possible that with further rounds of TTR, personalized key-target resizing would continue to increase accuracy.

Conclusion and Future Directions

In this paper, we introduced TTR with three text entry goals in mind: 1. Provide beneficial targeting practice, 2. Provide useful highlighting of areas for improvement, and 3. Generate beneficial training data for key-target resizing. Through a user study and simulation experiment, we demonstrated that the game indeed achieves these

three goals. As a future direction, we plan to explore using the game for real-time adaptation and collaborative filtering, as discussed in Section 2. We also plan to investigate adapting the text presented to users so that we can gather more data for areas in which our touch models have a significant amount of variance or where users simply need more practice. Finally, we are expanding the language coverage of TTR to enable widespread localization of our key-target resizing soft keyboard solution.

References

1. Amazon's Mechanical Turk. https://www.mturk.com
2. Appsfuse.com. http://www.appsfuze.com/games/windowsphone.boardandclassic/text-text-revolution,758, retrieved 3/13/2011.
3. Brewster, S., Chohan, F., & Brown, L. Tactile feedback for mobile interactions. Proc. of CHI, 159-162 (2007)
4. Chao, D. Doom as an interface for process management. Proc of CHI, 152-157 (2001).
5. Entertainment Software Association. Industry facts, http://www.theesa.com/facts/index.asp, retrieved 9/24/2010.
6. Evanini, K., Higgins, D., and Zechner, K. Using Amazon Mechanical Turk for Transcription of Non-Native Speech. Proc. of NAACL Workshop on Creating Speech and Language Data With Amazon's Mechanical Turk (2010)
7. Games with a Purpose, http://www.gwap.com/gwap/
8. Goodman, J., Venolia, G., Steury, K., & Parker, C. Language modeling for soft keyboards. Proc.of AAAI, 419-424 (2002)
9. Gunawardana, A., Paek, T. and Meek, C. Usability guided key-target resizing for soft keyboards. Proc. of IUI, 111-118 (2010)
10. Kölsch, M. & Turk, M. Keyboards without keyboards: A survey of virtual keyboards. Technical Report 2002-21. University of California, Santa Barbara (2003)
11. Hoggan, E., Brewster, S., and Johnston, J. Investigating the effectiveness of tactile feedback for mobile touchscreens. Proc. of CHI, 1573-1582 (2008)
12. MacKenzie, I. S. and Tanaka-Ishii, K. Text Entry Systems: Mobility, Accessibility, Universality. Morgan Kaufmann (2007)
13. Malone, T. M. What makes things fun to learn? Heuristics for designing instructional computer games. Proc. of SIGSMALL, 162-169 (1980)
14. Marge, M., Banerjee, S. and Rudnicky, R. Using the Amazon Mechanical Turk for transcription of spoken language. Proc. of ICASSP, March (2010)
15. Paek, T. and Hsu, B. Sampling representative phrase sets for text entry experiments: A procedure and public resource. In Proc. of CHI, (2011)
16. Rabin, E. & Gordon, A.M. Tactile feedback contributes to consistency of finger movements during typing. Experimental Brain Research, 155(3), 1432-1106 (2004)
17. SpeedType. http://itunes.apple.com/us/app/speedtype/id287255484?mt=8
18. Turbo Type – The typing game to type fast. http://itunes.apple.com/app/turbo-type-the-typing-game/id374229839?mt=8
19. von Ahn, L. Designing games with a purpose. Communications of the ACM, 51(8), 58-67 (2005)
20. von Ahn, L. and Dabbish, L. Labeling images with a computer game. Proc. of CHI, 319-326 (2004)
21. Windows Phone 7 Developer Tools. http://developer.windowsphone.com/windows-phone-7
22. Windows 7 Speech Recognition Tutorial. http://www.microsoft.com/enable/training/windowsvista/srtrain.aspx