SWIFT: A Short Word Solution for Fast Typing

Philippe Roussille and Mathieu Raynal(✉)

University of Toulouse - IRIT, Toulouse, France
{Philippe.Roussille,Mathieu.Raynal}@irit.fr

Abstract. In this paper, we study one specic problem linked to text input techniques based on prediction and deduction lists; namely, short-word problems. Indeed, while prediction is fast and can be easily made effective for long words (e.g. more than 4 characters), short words take longer to be typed with prediction: the time spent browsing a list-based interaction slows the user down. The present study compares two possible approaches where the user selects inside a prediction list of short words versus tactile exploration (native Voive Over for Apple). Results of our comparative study reveal that our technique reduces the overall typing time and the error rate by 38 % compared to tactile exploration.

Keywords: Text input · Visual impairment · Touchscreen · Mobile device · Short words · Word prediction

1 Introduction

Touch screens rapidly and significantly replace physical keyboards on mobile devices. Hence, text entry is now dependent on software (or virtual) keyboards that are widely used by sighted people, but raise accessibility issues for visually impaired users. These users rely on tactile exploration with vocal feedback of the whole screen for entering text, which is time consuming.

In order to solve this problem, different prediction-based techniques or deduction system exist. However, if these systems are generally enhance the speed of text-entry for users, they are inappropriate for entering short words. For example, within the DUCK [6] system, the use of the deduction system increases the input speed for words comprised of 5 characters or more. For shorter words, the use of the deduction system is too costly in time compared to the low number of characters to enter. This is even more a problem that some short words are very frequently used.

In this article, we therefore focus on entering short words. Our study aims to improve the typing of such words for use with the DUCK system. The proposed interactions have therefore been designed to be easily integrated and in tune with the rest of the interactions already proposed on this system. Both proposals interactions were evaluated and compared with the text input system traditionally used (like VoiceOver entry system).

DOI: 10.1007/978-3-319-41267-2-65

2 Related Works

2.1 Adaptation of the Traditional Keyboard

Text to speech (TTS) synthesis is the most used method to assist text entry on a mobile device coupled with a standard keyboard layout (e.g. QWERTY or AZERTY, depending on the culture). The underlying principle is simple: the user moves his finger onto the keyboard or hits a key, and the keys he hits or he moves across are spoken (see e.g. Apple VoiceOver). When he releases his finger, the corresponding character is inputted. Additional gestures can be used as well. One big problem with that comes from the fact that the keys of a virtual keyboard on a mobile device are very small. Thus, the risk of mistyping a letter is quite high. The second hindrance of such a solution resides in the need for users to go through most of the keys before finding the one they seek which has been reported as a "painful exploration" by the users because it usually takes some time. The offered sets of gestures might also be too complex to perform in mobility.

2.2 Specific Text Input System

More specific text-input system exist. They are often based on a character encoding which allows to limit the number of keys and thus help visually impaired people to find their location onscreen. The most known encoding for visually impaired people is the Braille alphabet. BrailleType [5] or TypeInBraille [4] adapted this encoding on touchscreens. In order to do so, such keyboards mimic a 6-dots Braille cell onto the phone screen. The characters are entered by successively activating the dots of the cell. Unlike the QWERTY standard keyboard, these keyboards have a much easier learning phase, due to the similarity with the Braille alphabet. On the other hand, the user must fill-in the dot matrix for each character, which leads to a quite high Keystroke Per Character (KSPC), hence drastically reducing text-entry speed.

Other methods, like NavTouch [2] & NavTap [3] allow the user to dynamically select a letter with a specific gesture performed anywhere on the screen, drawing a pattern on the device. These techniques are very efficient for quickly targeting one letter. However, they generate high cognitive load to simultaneously remember the correct gesture, as well as the position of the typed letter in the word.

The No-Look-Notes [1] text-entry method uses multiple fingers, relying on the multi-touch capacities of the device. The screen is divided in different zones, with a set of characters associated for each. The user can search for a character among a group of letters with one finger, and confirms the selection via a split-tapping. This method provides a fast and easy access to letters, but drastically increases the number of taps, which slows word typing down, and may cause muscular fatigue.

3 DUCK Keyboard

3.1 Principle

DUCK (deDUCtive Keyboard) [6] is based on typical existing keyboard layouts, so layouts are the most used ones, meaning that most people are acquainted with them. We decided, however, to simplify the user interaction to type letters. The main objective was to get rid of the accurate search of each character on the layout. Besides, we aimed to compensate for the lack of precision related to small and mobile keyboards.

The user initially explores the keyboard to find the first letter of the intended word. Each finger motion provides a vocal feedback, allowing the user to locate a letter on the layout. Once he releases the key, the corresponding character is selected. For typing the remaining letters of the word, the user doesn't have to explore the keyboard again. He just needs to press the location where he believes the keys are according to the memorized representation of the AZERTY or QWERTY layout.

The problem of such a solution relates to words shorter than four characters. Indeed, the interaction offered in the DUCK keyboard to confirm words is compensated for words that are long enough (words that contains more than five characters), but is inefficient when used for shorter words.

3.2 Short-Word Problems

Zhai and Kristensson [7] indirectly studied the short-words problem by using shorthand gestures for the most frequent short words of the English language. However, these gestures are based on the keyboard layout. A novice user can therefore use the visual keyboard representation to draw his gesture. This is not applicable for visually impaired people since they can't access that visual representation. Besides, it would be too hard for the user to memorize all the gestures.

This article aims to explore various interaction techniques in order to improve short-word typing by visually impaired people.

4 Proposed Interactions

As we saw, the DUCK keyboard is efficient to type words longer than five character. We are thus looking for an interaction technique which would be usable on the DUCK keyboard while allowing to type words shorter than 4 character faster than the used one.

In order to make such a technique easily integrable with DUCK, we kept the same interaction paradigms that DUCK typing uses:

– the user select the first character of the word he wants to type through a keyboard exploration;

– then, in order to access the most frequently used short words, he needs no additional key hits.

He just has to validate the first character of the word to access a list (of at most seven words) of the most frequently used short words starting with that character.

er diffrentes techniques d'interaction pour passer d'un mode l'autre: Our study is thus focusing on the choice of the best interaction to switch from classical input mode of a word with DUCK to typing a short word. As such, we chose to compare different interaction techniques to switch from one mode to another: In the first technique, which we called "dual-tap" validation, the user browses the keyboard by dragging his finger to locate the letter he looks for. He then releases his finger from the screen to select the character. The user then taps on the screen with two fingers to validate his choice and switch to the list mode. In the second technique, called "split-tap" validation, the character is selected the same way, but the user has to press a second finger without releasing the first finger. When he releases both fingers, the keyboard switch to the list.

4.1 Display and Feedback

In order to maximize the easiness of use for the user, we displayed the list in the same way as selecting a word with the system DUCK: over the whole screen, and each item is evenly taking the same size. They are contiguously aligned along the width of the screen, while taking the full height. When the user browses the list, he has to keep his finger pressed onto the screen, and the underlying elements are read at the same time. Releasing his finger validates the word underneath. Furthermore, in order to facilitate memorization, words are always displayed in the same order.

5 Method

5.1 Comparison Technique and Hypothesis

In order to correctly assess our techniques, we chose to compare both of these techniques to a third one, which keeps the tactile exploration principle: the user maintains his pressure as he slides his finger onto the next letters, and performs a split-tapping gesture by pressing a second finger on the screen in order to add a letter to the word hes typing. Such an approach also allow the user to increase his knowledge of the keyboard, while not being intrusive in the proper functioning of the device. We named that technique the "slide-typing" approach.

We selected this method as a basis for our comparison due to the fact that the study we made for DUCK [6], this technique was better than DUCK for words shorter than four characters.

Our main hypothesis is that by reducing the number of actions needed to type a short word, our interaction techniques will take less typing time; and will thus be faster than the currently used technique. Interactions are more precise than DUCK.

5.2 Participants

12 participants (4 women and 8 men, mean age = 24.7) with normal or corrected-to-normal vision participated in our study. They were all blindfolded to ensure they couldn't see the screen at all. They had a familiar use (from a daily use) of smart phones and mobile devices. Among them, six claimed to have a good knowledge of the AZERTY layout due to heavy gaming experience. Our choice of recruiting non-visually impaired participants is justified to verify the evaluated techniques as a proof of concept.

5.3 Apparatus

Participants used a Samsung Galaxy SIII smartphone. The device has a resolution of 306 ppp for a 136.6 mm × 70.6 mm screen, a ARM Cortex-A9 MPCore Quad core set at 1.4 GHz and uses Android 4.3. They used their finger to navigate through the items. The items were synthesized from text using the Google Translate service. The audio was trimmed to provide feedback as fast as possible, reducing any possible delay. Nothing was displayed onto the screen during the experiment, making the grids and items invisible to the users.

5.4 Procedure

The task was to type a short word (a word comprised of less than four letters) as fast as possible. The task went as follow: first, the participant was read the instruction, he was then given the word to search, then he had to type it. If he answered a different word, we considered his answer to be wrong. Once a session was completed, the participant was asked to answer a SUS test for each technique, and state his preference among the interactions he went through by ranking them in the order of preference.

5.5 Design

Each participant had to go through three sessions, one for each interaction: dual-tap validation, split-tap validation, slide-typing. All three sessions were run one after the other. The twelve participants were then split at random into three even groups. The groups were then assigned interactions according to a balanced Latin square. For every session, the participant had to go through 44 tasks. For each task, the word to select was picked at random. However, in each series, every initial letter is checked with two different words.

This leads us to a total of 1,584 trials.

5.6 Collected Data

In order to study the different presentations, we collected various data during the experiment: we collected the number of items seen by the user, the time taken for the participant to make his selection, the errors he made and the answers he gave to the SUS questionnaire.

6 Results

6.1 Time Spent

We decided to measure the time taken by the user by counting between the moment he entered the first letter of the word, and the moment he validated his word.

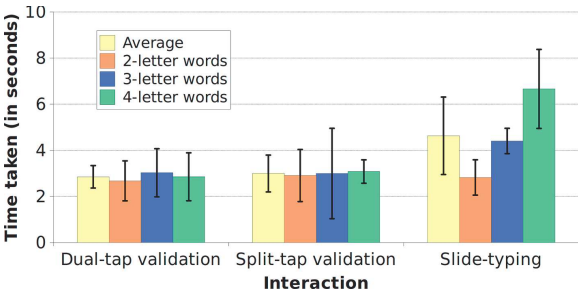

Fig. 1. Time taken to type a word by technique and by word length (Color figure online)

As we see on Fig. 1, for both list-based techniques, there is no significant ($p > 0.05$) influence on the word length: the dual-tap validation takes on average 2.85 s compared to the 2.99 s for the split-tap validation.

The time taken to type a word using the slide-typing is dependent on the word length ($p < 0.04$), and it increases along the length of the word (2.82 s for two letter words versus 6.66 s for four letter words).

Nevertheless, it is quite apparent that both list-based techniques are significantly faster ($p < 0.05$) than the slide-typing technique (2.85 s and 2.99 s versus 4.63 s on average). The list-based techniques have no significantly different typing speed. They are also operating in near constant time. In comparison, the DUCK keyboard requires 7 s on average to type words under five characters, which is longer than our techniques here. This validates our main hypotheses.

6.2 Error Rate

We then computed the error rate for each technique. It corresponds to the percentage of the wrongly selected words during the input over the total of these words.

As seen on Fig. 2, the two techniques based on the lists offer a good accuracy with 13 % for dual-tap validation and 11 % for split-tap validation, without any significant effect due to the while the slide-typing technique offers a 21 % error rate. DUCK itself offers 20 % error rate.

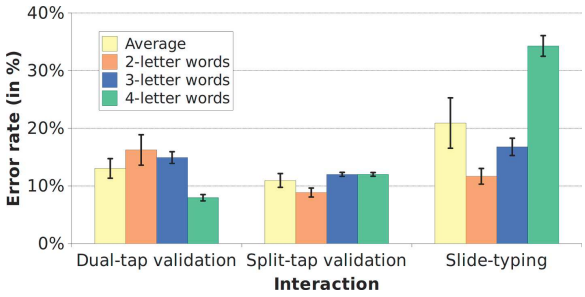

Fig. 2. Error rate when typing a word by technique and by word length (Color figure online)

6.3 User Feedback

We computed the SUS score obtained for each technique given the users' replies to the test. The slide-typing technique scored 86, just before the dual-letter validation technique which scored 85. The split-tap technique scored last with 79. When asked to rank the techniques, the users followed what they thought on the SUS assessment: the dual-tap validation technique was ranked first (5 our users), while the slide-typing approach came second; the split-tap validation technique being the most disliked interaction.

7 Discussion

Given our results, we can safely affirm that word lists are really effective when used in a short-word typing context. Such techniques are more efficient than the existing method built-in DUCK.

As we found in the experiment, the lists are really effective in a short word typing context. They turned to be faster than the slide-typing method, as well as being more resilient to errors. The lists work in near constant time. When it comes to accuracy, lists retrieve the words with less errors. This can be explained by the fact that both of the lists offers the possible word without any mistake in its spelling. As such, the only possible error in the validation is selecting a completely different word; while such errors are possible at each letter if the user is entering a word letter by letter (by using the slide-typing technique).

Besides, as the users knowledge of the list increases with its use, he spends less time browsing the list seeking the item he wants, and could validate his answer directly.

However, lists are very static. Indeed, the users can't choose other words than the words that have been elected to be in the list.

Therefore, our technique is very effective on the most commonly used short words, but is useless for other short words. However, with a maximum of 7 words starting with the same letter, our system allows the use of 124 most used words short, representing 74 % of the short words used in the French language.

Finally, word lists are static, we believe that the performance obtained in this study without learning can be improved gradually as the user will type words

using these lists. Indeed, the more he will use them, the more he will know the position of the words in the list and will lose less and less time to browse through the list to find the desired word. On the other hand, he will also know if the word is in the list of suggested words, and therefore may or may not use this interaction to quickly enter a short word.

8 Conclusion

The list-based solution we offer is faster and more accurate than a technique based on tactile exploration, or the technique offered in DUCK. Even if this solution is limited to a restricted number of short words, it will improve the overall typing speed as these words are the most frequently used ones.

References

1. Bonner, M.N., Brudvik, J.T., Abowd, G.D., Edwards, W.K.: No-look notes: accessible eyes-free multi-touch text entry. In: Floréen, P., Krüger, A., Spasojevic, M. (eds.) Pervasive 2010. LNCS, vol. 6030, pp. 409–426. Springer, Heidelberg (2010)
2. Guerreiro, T., Lagoá, P., Nicolau, H., Gonçalves, D., Jorge, J.A.: From tapping to touching: making touch screens accessible to blind users. IEEE MultiMed. **15**(4), 0048–0050 (2008)
3. Guerreiro, T., Nicolau, H., Jorge, J., Gonçalves, D.: Navtap: a long term study with excluded blind users. In: Proceedings of the 11th International ACM SIGACCESS Conference on Computers and Accessibility, Assets 2009, pp. 99–106. ACM, New York (2009)
4. Mascetti, S., Bernareggi, C., Belotti, M.: TypeInBraille: quick eyes-free typing on smartphones. In: Miesenberger, K., Karshmer, A., Penaz, P., Zagler, W. (eds.) ICCHP 2012, Part II. LNCS, vol. 7383, pp. 615–622. Springer, Heidelberg (2012)
5. Oliveira, J., Guerreiro, T., Nicolau, H., Jorge, J., Gonçalves, D.: BrailleType: unleashing braille over touch screen mobile phones. In: Campos, P., Graham, N., Jorge, J., Nunes, N., Palanque, P., Winckler, M. (eds.) INTERACT 2011, Part I. LNCS, vol. 6946, pp. 100–107. Springer, Heidelberg (2011)
6. Roussille, P., Raynal, M., Jouffrais, C.: Duck: a deductive soft keyboard for visually impaired users. In: Proceedings of the 27th Conference on L'Interaction Homme-Machine, IHM 2015, pp. 19:1–19:8. ACM, New York (2015)
7. Zhai, S., Kristensson, P.: Shorthand writing on stylus keyboard. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI 2003, pp. 97–104. ACM, New York (2003)