Semantic Keyboard: Fast Movements between Keys of a Soft Keyboard

Mathieu Raynal[1], I. Scott MacKenzie[2], and Bruno Merlin[3]

[1] IRIT – ELIPSE Team, University of Toulouse, Toulouse, France
mathieu.raynal@irit.fr
[2] Department of Electrical Engineering & Computer Science, York University, Toronto, Canada
mack@cse.yorku.ca
[3] Universidade Federal do Pará, Cametá, Brasil
brunomerlin@ufpa.br

Abstract. In this paper we describe Semantic Keyboard: a soft keyboard augmented by semantic pointing. The cursor crosses faster over keys containing low-probability letters (considering the prefix already entered). This optimization reduces the movement of the pointer by 60%, and increases the text entry speed by 13.5% after the first character in a word. Accuracy is equivalent to a regular soft keyboard.

Keywords: Soft Keyboard, Text Entry, Character Prediction.

1 Introduction

Highly used since the emergence of mobile phones with touch screens, soft keyboards were originally designed to enable people with motor disabilities to access computers. The basic soft keyboard imitates a physical keyboard. Motor-disabled people use it with an adapted pointing device. However, the use of a single pointer slows text input. In addition, repetitive movements between keys tire the user quickly.

To solve these problems, several solutions have been explored. Many take advantage of letter co-occurrence (aka digram) statistics in the language. They typically lead to a static or dynamic reorganization of the keyboard or other improvements considering co-occurrence. A well known technique is to provide a completion list [2],[7] to reduce the number of actions to input a word. However, every solution is a compromise: New layouts are hard to learn; dynamic changes increase cognitive load and completion lists are often used inefficiently.

Our goal is to dynamically exploit language statistics to improve typing performance without adding visual changes that disturb the user. The aim is to reduce key pointing time while maintaining the accuracy of pointing. Our system is useful when using a soft keyboard with any device that manipulates a pointer on the screen. Consequently, our main end users are motor-impaired persons. In this report, we present our system and an initial evaluation with able-bodied users which confirms our hypothesis. A use-case with end users is ongoing; those results will be presented in a final report.

K. Miesenberger et al. (Eds.): ICCHP 2014, Part II, LNCS 8548, pp. 195–202, 2014.

2 Use of Character Prediction Systems

Entering text with a standard soft keyboard through a single pointing device corresponds to pointing to keys one by one. The task can be modeled by Fitts' law [4]. Toward this, two solutions are possible to increase text input speed: reducing the distance between the keys or expanding the keys.

The first solution involves switching the layout to reduce the distance between the most frequent co-occurrences [6],[9],[13]. However, a new letter arrangement imposes a learning period which often discourages the user. Therefore, this solution is rarely used.

Also, while this solution works well for common co-occurrences, it is less effective for infrequent co-occurrences. To improve optimization whatever the input letter, it is important use a dynamic prediction system. In 80% of cases, the desired letter is among the four most probable to be typed [10]. This statistic increases as the length of the prefix increases: Likely letters are more numerous and therefore more predictable. Two strategies simplify pointing for the most probable letters. The first is to dynamically change the letter arrangement (key position) and size during input [1],[8]. The second is to introduce additional keys near the pointer [10] that contain the most likely letters.

However, these solutions have limits: Dynamic changes in the layout induce a cognitive cost. It is tiring for the user and it adversely impacts performance. So, in the end, input speed is not improved with these keyboards. Similarly, when new GUI elements appear, visual scanning time increases and text input speed is reduced.

Consequently, to benefit from prediction without changing the appearance of the standard keyboard, we propose to dynamically modify the pointer speed according to the probability of typing the crossed-over keys.

3 Semantic Keyboard

3.1 Principle

Our Semantic Keyboard is based on the paradigm of semantic pointing [3]: that is, separating the visual space and motor space. The interface retains the same visual appearance. However, every object occupies a different area in the motor space. The size of the area depends on the object's importance in the interaction context. In our Semantic Keyboard, visual space is the soft keyboard and motor space is represented through the pointing device. Keys bearing letters with low probability get less emphasis in the motor space: The pointer will quickly cross over them. Conversely, keys bearing letters with high probability get more emphasis in the motor space: The cursor dwells longer.

Specifically, the idea is to accelerate the cursor when it passes over keys displaying letters with low probability and, reciprocally, to lower the cursor velocity when passing over keys containing letters with high probability. Thus, the user should more quickly access letters of interest while still maintaining the possibility of inputting other letters. But, at the same time, the visual aspect of the keyboard remains unchanged.

3.2 Implementation

Our Semantic Keyboard uses a character prediction system based on a lexical tree [2]. When the user enters a prefix, the prediction system classifies letters by their probability of occurrence and then associates an enlargement coefficient with each key. With the enlargement coefficient, the pointer will accelerate or slow down.

Each key K_i has a coefficient C_i between 1 and N. N is the maximum value of acceleration. C_i is calculated from the character frequency F_i of the key K_i. The character frequency F_i is calculated from the prefix already entered:

$$C_i = F_i \times N \text{ and } \sum_{i=1}^{NB} F_i = 1, \text{with } NB \text{ the number of keys} \tag{1}$$

When the pointer passes over the keyboard, it may cross neighboring keys with different probabilities. To avoid dramatic speed variations, the acceleration coefficient applied to the pointer depends on the position of the pointer on the key. If the pointer is close to another key, the coefficient is calculated according to the coefficient of the closest keys and the distance separating the center of neighboring keys and the pointer. D_i is the distance between the pointer position and the center of the nearest key. W_i is the size of the key:

$$C = \begin{cases} C_i & \text{if } D_i < MAX \\ \frac{\sum_{j=1}^{NB\_Keys} D_j \times C_j}{\sum_{j=1}^{NB\_Keys} D_j} & \end{cases} \tag{2}$$

NB_Keys is the number of the nearest keys included in the calculation and MAX the maximum distance from the center of the key on which the coefficient remains unchanged. After this maximum distance, the coefficient is weighted by the coefficients of nearby keys.

Thanks to this coefficient, the pointer position is:

$$P = P_{old} + ((P_{new} - P_{old}) \times C \times Speed) \tag{3}$$

P_{old} is the old pointer position and P_{new} is the new one before the addition of the acceleration.

After each character input, the probabilities of character occurrence are recalculated and keys are resized (in the motor space).

4 Method

4.1 Hypothesis

To test our Semantic Keyboard, we conducted an experiment comparing the Semantic Keyboard with a typical AZERTY keyboard. The two main hypotheses were that semantic pointing would decrease the distance travelled by the cursor of the pointing device and lightly improve user performance. We define performance as text input speed and accuracy.

4.2 Participants

Twelve able-bodied participants, two females and ten males, took part in the experiment. They ranged in age from 21 to 44 ($mean = 28.6, SD = 6.15$). All were volunteers, right-handers, and computer specialists. All participants were regular users of desktop computers and were acquainted with pointing devices.

4.3 Apparatus

The experiment was conducted using a Dell laptop with 2.5 GHz speed and the Microsoft Windows 7 operating system. Participants interacted with the soft keyboard through a mouse. Keyboard layouts were restricted to the 26 characters of the Latin alphabet and the space bar. The soft keyboard was developed in Java SE 6.

4.4 Procedure

Participants entered 21 sentences with the both keyboards. They were instructed to enter the sentences as quickly as possible. The sentence to copy was presented on a line, and the text input by the participant appeared on the line below. Text entry errors were not displayed on the screen. Instead, there was visual and auditory feedback signaling the error. The cursor did not move until the participant entered the correct character. At the end of each sentence, participants hit the Space bar. After the experiment, participants were asked to complete a questionnaire soliciting demographic information and impressions on the both keyboards.

4.5 Design

A repeated-measures design was used. There was a single factor, keyboard, with two levels: AZERTY and Semantic Keyboard. Participants were randomly assigned to two groups of six. In the first group, participants began with the AZERTY keyboard and ended with the Semantic Keyboard. The order was counterbalanced by groups. For each keyboard, participants entered 21 sentences. The sentences were chosen randomly from a set of 50.

The sentences contained common words and were statistically representative of the participants' language, which was French (respecting the frequency of bigrams and trigrams). The dependant variables were the distance travelled by the cursor, text entry speed, and accuracy.

5 Results and Discussion

A statistical analysis showed that the order in which the exercises were performed had no impact on the results ($F_{1,10} = 0.059,\ p = .81$). Thus, counterbalancing had the desired effect.

5.1 Distance

The distance covered by the pointer in the motor space (in pixels) was computed for both keyboards. The results show that distance was 60% less with the Semantic Keyboard than with the AZERTY keyboard. The mean distance covered by the pointer to point to a key was 225 pixels with AZERTY compared to 90 pixels with the Semantic Keyboard. The difference was statistically significant ($F_{1,10} = 2721.5,\ p < .001$).

5.2 Entry Speed

Entry speed was computed two ways. First, we computed the text entry speed per sentence. This was calculated by dividing the length of the sentence (including the space character between words and at the end of a sentence) by the time (in seconds) to enter this sentence. Finally, this speed in characters per second (cps) is multiplied by sixty and divided by five to obtain the speed in words per minute (wpm) [12].

The input speed was almost equivalent for the two keyboards: 10.30 wpm versus 10.43 wpm ($F_{1,10} = 0.223,\ p = .65$). Interviews with participants after the experiment helped to identify potential issues: Several participants described being confused by the Semantic Keyboard at the beginning of words. Indeed, the prediction system predicts the probability of each character after a prefix is already entered. Thus, for the first character, Semantic keyboard offers no acceleration factor, which bothered some participants.

To verify this, entry speed was recalculated taking into account only the words. Entry time was calculated between the first character of the word and pressing the Space bar at the end of the word. The number of characters here is the length of the word. With this method of calculation, the entry speed was 13.10 wpm for the Semantic keyboard. This is 14% higher than the rate of 11.54 for the AZERTY keyboard. The difference was statistically significant ($F_{1,10} = 21.12,\ p = .005$).

The average word entry speed shows that Semantic Keyboard works well when the system predicted the most likely characters. Figure 1 shows the entry speed by sentence. The effect of the Semantic Keyboard is immediate. The Semantic Keyboard does not require learning, even if the improvement grows lightly during the first sentences.

5.3 Accuracy

During entry, when the current character differed from the expected character, an error was recorded. The error rate was obtained by dividing the number of errors by the number of characters. The number of errors was computed using the MSD method [5],[11]. The error rate was 1.3% for the Semantic Keyboard and 0.95% with the AZERTY keyboard. This difference was not significant ($F_{1,10} = 4.204,\ p = .065$). Overall, error rates were below 3% which is generally acceptable for text input.

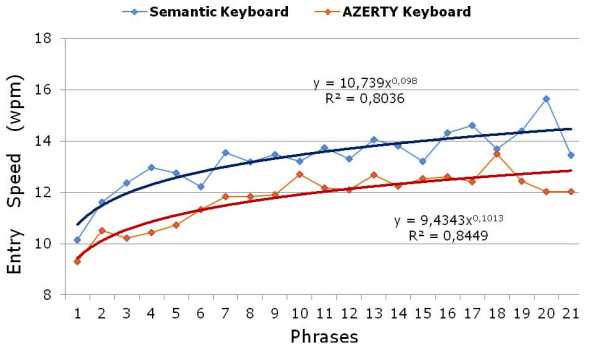

Fig. 1. Entry speed by keyboard and phrases

5.4 User Satisfaction

Responses were rated on a 7-point Likert scale, with 1 the least favorable response and 7 most favorable. The questionnaire solicited responses about comfort, fatigue, effort, accuracy and speed perception for the both keyboard (see Fig. 2).

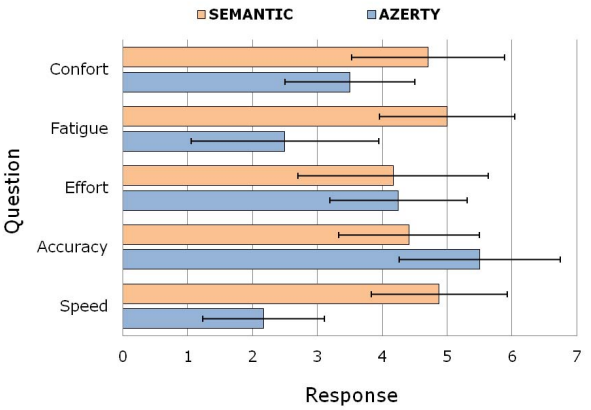

Fig. 2. Results of the questionnaire. A response of 7 is most-favorable, and 1 least-favorable.

The opinion of participants corroborates the quantitative results. We observed the following: On the one hand, the Semantic Keyboard is faster than the AZERTY keyboard. On the other hand, the AZERTY keyboard is more accurate than the Semantic Keyboard. Another important finding is that participants

rated the Semantic Keyboard less tiring than the AZERTY keyboard ("Fatigue" in Fig. 2). This information is important because motor-impaired users, who are often forced to use a soft keyboard, often suffer from fatigue during text input.

6 Case Study with Motor-Disabled Person

To verify the relevance of these initial results for the target population, we conducted a case study with a motor disabled person suffering from muscular dystrophy. The subject conducted the same experiment as the able-bodied participants but with fewer phrases to copy because of anticipated fatigue.

We draw two conclusions from the case study: First, the participant had more difficulty to handle the mouse pointer. As a result, he traveled more distance with the mouse pointer to type a word: With AZERTY keyboard, the movement distance was 35% less for able-bodied persons than the motor-disabled person (225 pixels versus 346). Therefore, he entered text slower than the able-bodied participants (6.58 wpm versus 11.54 wpm).

The other important observation is that the results obtained in the first experiment were confirmed in the case study. Indeed, the distance travelled by the pointer decreases 30% using Semantic Keyboard (244 pixels). This decrease causes an increase of the text input speed of 15%. The speed with the Semantic Keyboard was 7.59 wpm and with the regular keyboard 6.58 wpm.

7 Conclusion

The Semantic Keyboard presents a technique that uses character prediction to alter the cursor speed without changing the appearance of the keyboard. It has the advantage of character prediction without disturbing the user. The technique reduces the distance covered by the cursor but does not reduce the input time. In an experiment, the entry speed was about 14% higher after the first character in a word for the Semantic Keyboard compared to an AZERTY keyboard.

The experiment showed that the semantic pointing could work well in concert with a robust character prediction system. Since the Semantic Keyboard can be effective for text input, we expect to improve the prediction system so it can also be used at the beginning of the word. As well, the system should consider accents and punctuation in order to offer a comprehensive keyboard suitable for everyday tasks.

References

1. Aulagner, G., François, R., Martin, B., Michel, D., Raynal, M.: Floodkey: Increasing software keyboard keys by reducing needless ones without occultation. In: Proc. ACS 2010, pp. 412–417. WSEAS (2010)
2. Badr, G., Raynal, M.: Evaluation of wordTree system with motor disabled users. In: Miesenberger, K., Klaus, J., Zagler, W., Karshmer, A. (eds.) ICCHP 2010, Part II. LNCS, vol. 6180, pp. 104–111. Springer, Heidelberg (2010)

3. Blanch, R., Guiard, Y., Beaudouin-Lafon, M.: Semantic pointing: Improving target acquisition with control-display ratio adaptation. In: Proc. CHI 2004, pp. 519–526. ACM (2004)
4. Fitts, P.M.: The information capacity of the human motor system in controlling the amplitude of movement. Journal of Experimental Psychology 74, 381–391 (1954)
5. Levenshtein, V.: Binary codes capable of correcting deletions, insertions and reversals. Soviet Physics Doklady 10, 707 (1966)
6. MacKenzie, I.S., Zhang, S.X.: The design and evaluation of a high-performance soft keyboard. In: Proc. CHI 1999, pp. 25–31. ACM (1999)
7. Masui, T.: An efficient text input method for pen-based computers. In: Proc. CHI 1998, pp. 328–335. ACM (1998)
8. Merlin, B., Raynal, M.: Evaluation of SpreadKey system with motor impaired users. In: Miesenberger, K., Klaus, J., Zagler, W., Karshmer, A. (eds.) ICCHP 2010, Part II. LNCS, vol. 6180, pp. 112–119. Springer, Heidelberg (2010)
9. Raynal, M., Vigouroux, N.: Genetic algorithm to generate optimized soft keyboard. In: Extended Abstracts Proc. CHI 2005, pp. 1729–1732. ACM (2005)
10. Raynal, M., Vigouroux, N.: KeyGlasses: Semi-transparent keys to optimize text input on virtual keyboard. In: Proc. AAATE 2005, pp. 713–717. IOS Press (2005)
11. Soukoreff, R.W., MacKenzie, I.S.: Measuring errors in text entry tasks: An application of the Levenshtein string distance statistic. In: Extended Abstracts Proc. CHI 2001, pp. 319–320. ACM (2001)
12. Yamada, H.: A historical study of typewriters and typing methods, from the position of planning Japanese parallels. Journal of Information Processing 2, 175–202 (1980)
13. Zhai, S., Hunter, M., Smith, B.A.: The Metropolis keyboard: An exploration of quantitative techniques for virtual keyboard design. In: Proc. UIST 2000, pp. 119–128. ACM (2000)