Performances of Multiple-Selection Enabled Menus in Soft Keyboards

Gennaro Costagliola, Vittorio Fuccella
Department of Mathematics and Informatics
University of Salerno
{gencos,vfuccella}@unisa.it

Michele Di Capua, Giovanni Guardi
Unlimited Software
md@unlimitedsoftware.it, g.guardi@u-s.it

Abstract

In this paper we present a new text entry approach based on soft keyboards and a study on its expected performances. The basic idea of the approach is to enhance the keyboard with a menu whose items (characters) can be selected more than once through a single pointer (pen, finger, etc.) stroke, in order to enter particularly frequent text patterns. The gesture to enter these patterns is similar to that used for Cirrin. Furthermore, using well-established methods for measuring the time required for performing simple interactions with pointers on touchscreens, we have evaluated the expected performances of our method. The results, obtained through a text entry simulation, are encouraging with respect to those measured for previous approaches based on menu-augmented keyboard.

1. Introduction

Fast text entry in mobile devices is still a challenging issue. Soft keyboards on touchscreen are still largely employed on palmtop and smart phones. Soft keyboards enable text input through *tapping*. Each *tap* corresponds to a single character input. Even though more efficient keyboard layouts exist [10, 18], the QWERTY layout is still the most used, due to the users' familiarity acquired in the use of desktop and laptop computer keyboards. Other layouts do not meet a wide audience, probably due to the difficulty to become familiar with an unknown layout [11]. In order to accelerate text entry, menu augmented keyboards [5] have been introduced. They aim at improving text entry performances by enabling text entry through *flicks*, in addition to the classical *tap*. A *flick* corresponds to the input of a digraph in which the first letter is identified by the starting point of the interaction, the second by its direction.

In this paper we present a new text entry approach on menu-augmented soft keyboards and a theoretical study on its performances. The approach is based on the following idea: the menu items can be selected more than once, in order to enter particularly frequent text patterns with a single pointer stroke. The interaction to enter a text unit, by selecting a sequence of menu items, is similar to that used for *Cirrin* [12]. Users familiar with a given keyboard layout must not learn a new one: they just have to acquaint themselves with the new type of interaction.

Furthermore, the method has the advantage to significantly reduce the number of strokes (taps and gestures) necessary to enter text. Compared to past methods [5], which only enable input of digraphs, it enables the input of longer string chunks with a single stroke. It is worth noting that the user can choose whether showing the menu or not, eventually switching from *novice* to *expert mode*.

The paper also presents a theoretical analysis on the expected performances in terms of speed obtainable by an ideal user. The analysis model uses Fitts' Law [3] and Hick-Hyman Law [4] for evaluating performances for novice and expert users on the the QWERTY keyboard layout. Based on this model we built a text entry simulation and measured the expected performances of our approach showing that these are slightly better than those measured for previous approaches based on menu-augmented keyboard.

The rest of the paper is organized as follows: the next section contains a brief survey on text entry methods with soft keyboards, focusing mostly on the methods related to ours, and summarizes some basic concepts about performance analysis; section 3 gives an explanation of the proposed approach; in section 4 we present the analysis on the expected performances; lastly, some final remarks conclude the paper.

2. Related Work

2.1. Entry Methods

The spread of mobile computers and smart phones equipped with touch screens has attracted the interest of researchers on the problem of text entry in such devices. Several methods for accelerating the text entry task have been proposed. Some of the main directions followed by researchers are:

- proposal of *more efficient* keyboard layouts;
- proposal of interaction types minimizing the number of strokes for text entry.

In the former case, the research is mainly based on the idea that the keyboard layouts should minimize the distance between characters with a high probability of being consecutive in the words of target languages. To this aim, the frequency of digraphs in target languages (e.g. English) has been analyzed in texts and reported in tables [15]. The studies have resulted in the proposal of specific layouts, such as OPTI [10], Fitaly [16], Atomik [19] and Metropolis [18].

In the latter case, methods enabling text entry through interaction types different from *tapping* have been proposed. An example is the selection of pie menu items. Among these methods, the earliest, such as *T-Cube* [17], enabled input of single characters per menu item selection, through a *flick* interaction. Then, researchers noticed that enabling the menu item selection on a virtual keyboard it is possible to input digraphs through a single stroke. The first character is located on the keyboard and is identified by the starting point of the stroke. The second is located in the menu item and is identified by flick's direction.

Other methods enable the input of single characters with a single stroke without the use of a virtual keyboard (*unistroke* alphabets, such as *Graffiti* [2]), or enabling the input of entire words through a single stroke (word-level unistroke) on ad-hoc designed keyboard layouts. Among these, we can mention *Cirrin* [12] and *Quikwriting* [14]. In the above methods, the objective is pursued by arranging the keys in a special layout which facilitates the input of whole words without ambiguity. More recent methods enable the use of word-level unistroke still keeping keyboard layouts familiar to the user, as QWERTY. In those cases, e.g. in [6] and [20], it is necessary the application of sketch recognition and dictionary-based disambiguation methods.

We will now briefly describe *Cirrin*, since our method uses a similar interaction to enter text. *Cirrin* is an intuitive word-level unistroke text entry method on virtual keyboard proposed by Mankoff and Abowd in 1998. The keys are arranged in a circle, as shown in figure 1. In order to enter a word, the user, beginning from the middle of the circle, simply traces out a path that crosses the circumference at points corresponding to the characters of the word, in the right order. A space character is automatically inserted as soon as the pen is lifted. The arrangement of the letters in the circle is such to minimize the average distance the pen travels to write a word. *Cirrin* has shown itself to be about as fast as classical QWERTY virtual keyboards and is particularly suitable for mobile computers.

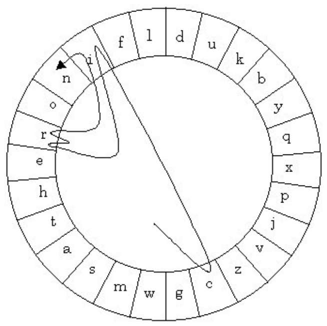

Figure 1. The Cirrin text entry method.

2.2. Performance Analysis

Performances of a text entry method can be evaluated from the points of view of speed and accuracy. Speed is measured in terms of *characters per second* (*cps*) or, more frequently, of *words per minute* (*wpm*). A word is defined as a sequence of five characters (regardless of whether they are letters, punctuation or spaces), thus we can convert *cps* in *wpm* by multiplying *cps* by 12. Accuracy can be measured as the percentage of errors in typing a phrase out of the number of characters of the phrase. For a more detailed insight on performance analysis, the reader should refer to [9].

While accuracy can be only measured in trials, speed can also be estimated with theoretical models. The models can be applied to measure both the novice and the expert performances. For the latter users, who are supposed to already know the keyboard layout, only movement efficiency must be considered, while for the former users, also the time needed for visual exploration to search for the destination key (or menu item) should be taken into account. To predict movement efficiency, Fitts' Law [3] is used. According to the Shannon formulation, [8], of the law, for movement along a single dimension, the average time T to complete the movement can be calculated as:

$$MT = a + b * log_2(D/W + 1) \quad (1)$$

where:

- a represents the start/stop time of the device and b stands for the inherent speed of the device. These constants can be determined experimentally by fitting a straight line to measured data.
- D is the distance from the starting point to the center of the target.
- W is the width of the target measured along the axis of motion.

To predict the Reaction Time (the time required to make a choice among n items), Hick-Hyman Law [4] is used. The law is expressed as follows:

$$RT = a' + b' * log_2(n) \tag{2}$$

where n is the number of items to choose from. The coefficients a' and b' are slope and intercept constants, similar to the coefficients a and b in Fitts' law.

3. The approach

The approach is an improvement of the classical stylus keyboarding. The improvement lays in enabling gesturing on the keyboard, besides the classical *tapping* interaction: the gesture is a drag of the pointer inside a menu containing a restricted set of frequent characters, one per menu item. The menu is shown as soon as the pointer is pressed on a key and disappears as soon as the user releases the pointer. The menu is displayed around the pressed key: menu items are arranged one per key side. More precisely, the internal perimeter of the menu coincides to the external perimeter of the key. For instance, if the character has a squared (or rectangular) shape, the menu will allow four different character choices.

While the menu is shown, it is possible to sweep out a gesture that touches only the desired characters in succession by dragging the pointer, without lifting it. This gesture is almost the same as for the *Cirrin* method. In *Cirrin* a space character is always automatically inserted at the end of the gesture, since the word is supposed to be completed with just one stroke. Since in our method text entry units are not always complete words, the space character can be inserted or not, according to the final position of the pointer when it is released: the adopted convention is to add a space at the end of the text unit only if the pointer is lifted after returning inside the character key area. Otherwise, if the pointer is lifted in the menu area, the string finishes with the last selected character. Note that in all the other cases the space bar is directly used.

Let us consider a soft keyboard augmented with a menu containing n characters $x_1, \ldots, x_n$. With our method, with a single stroke we can enter a text unit described by the following regular expression (for a short reference manual on regular expressions, see [13]):

$$.[x_1 x_2 \ldots x_n] + [\ ]? \tag{3}$$

The above pattern matches any text unit starting with any character (specified in (3) by the starting '.') chained to a sequence of one or more of the $x_1 x_2 \ldots x_n$ characters, (specified by $[x_1 x_2 \ldots x_n]$+) eventually ending with a space character, ([ ]?).

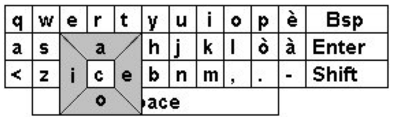

Figure 2. The QWERTY keyboard layout augmented with a menu.

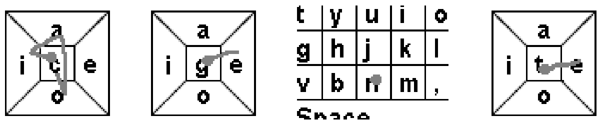

Figure 3. The 4 strokes needed to enter the text 'ciao gente'.

Our approach can be instantiated by associating each of the vowels 'a', 'e', 'i' and 'o' to one of the sides of the squared character keys. This choice is supported by the fact that these vowels are more easily remembered by users and that they are among the most frequent letters in many languages.

Arranging the menu items as shown in figure 2, the interaction sequence necessary to entry the Italian text *ciao gente* (*hello folks*, in English) is shown in figure 3. The string is ten characters long but it can be entered with a sequence of four strokes (taps or gestures). The strokes correspond to the input of the following sequence of text units $\{$*ciao* $\}\{$*ge*$\}\{$*n*$\}\{$*te*$\}$. Three text units out of four are entered through a gesture and only one through a *tap*. With the first stroke we can enter up to five characters.

Generalizing, the method allows to select a small number of frequent characters (4 with square keys, 6 with hexagonal keys and 8 with octagonal keys). The method can be instantiated differently for each specific language. In particular, the following parameters should be chosen appropriately:

- The number of menu items should be the result of a compromise: the highest this number, the highest the frequency of the match of the regular expression in (3) is. Conversely, the smaller this number, the faster the learning of the method by novice users is.

- The choice of the letters associated to menu items should take into account the frequency of the matches of the regular expression shown in (3), and users' learning preferences (vowels could be remembered more easily).

4. Expected performances

In this section we describe the theoretical analysis we have carried out on the expected performances in terms of speed obtainable by ideal novice (lower bound) and expert (upper bound) users. The users are supposed to be familiar with the keyboard layout. The performances have been measured for a user writing in Italian language. We expect similar results for other Romance languages, since they have similar letter and pattern frequency. The proposed evaluation model is an extension of the one used by Soukoreff and MacKenzie in [15] for evaluating virtual keyboard performances.

4.1 Reaction and movement time estimation

Since our approach enables two kinds of stroke (tapping and gesturing), our model separately evaluates the time needed for each of them. The former is the classical stroke used to enter a character located on the keyboard, while the latter is used to enter a text unit whose characters are located in the menu. The time required to perform each of them is calculated as follows:

Tapping. Since we assume that the users are already familiar with the keyboard, the required time is only given by the time needed to move the pointer from the previous key to the current one. We calculate the time to enter a c character located on the keyboard with the following formula:

$$Tk(c) = MT(c) \tag{4}$$

Where $MT(c)$ is the movement time to reach character c measured with Fitts' Law shown in (1).

Gesturing. In this case we consider both the novice and the expert performances. The time required for experts is only given by the time needed to drag the pointer among menu items, while for novices, we have to add the time necessary to visually scan the menu. This value is added every time a new menu item must be selected to enter the corresponding character, except for characters already selected in the same gesture. Summarizing, we calculate the time to enter a tu text unit using the menu with the following formula:

$$Tm(tu) = \begin{cases} MT(tu) & \text{if experts} \\ MT(tu) + RT(tu) & \text{if novices} \end{cases}$$

Where $MT(tu)$ is the movement time to enter the text unit tu, measured with Fitts' Law shown in (1); $RT(tu)$ is the reaction time (visual scan of the menu) to enter the text unit tu, measured with Hick-Hyman Law shown in (2).

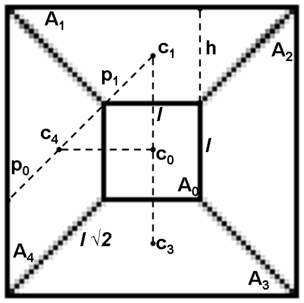

Figure 4. A model of menu with 4 items.

$MT(c)$ can be calculated with Fitts' Law. In particular, we use the following values for parameters:

- for intercept and slope, we use the values experimentally obtained by [7] for pointing tasks on tablet devices: $a = -55$; $b = 204$.
- for distance A, we use the distance between the centers of the previously pressed key and c;
- for W, the width of the target, we use the length of the side of the squared key.

It is worth noting that the measured time does not depend on the size of the keyboard, since both A and W grow proportionally with it.

$MT(tu)$ is calculated as the sum of the times to enter the single characters in tu. Supposing that tu is the concatenation of n characters $x_0 \ldots x_n$, then $MT(tu) = MT(x_0) + \ldots + MT(x_{n-1})$.

In particular, the movement time for the first character of the text unit, $MT(x_0)$, is calculated as done above for $MT(c)$, since it is located in the keyboard. The calculation of the time to select a character corresponding to a menu item $MT(x_i)$, with $1 \leq i \leq n-2$, i.e., is more complex and requires the analysis of the different movement types that can occur when dragging the pointer inside the menu.

Figure 4 shows a model of a key with a four items menu. We will refer to the internal area of the key with A_0, its center with c_0; the upper, eastern, lower and western areas are referred to with A_1, A_2, A_3 and A_4, their centers with c_1, c_2, c_3 and c_4. The key area is squared, while that of menu items has the shape of an isosceles trapezoid, whose smaller base coincides with the key side and whose base angles are 135 degrees. We also assume that the height of the trapezoid has the same length of the key side. The first movement, required to select the second letter of the text entry unit, is a drag from the center of the key to the center of a menu item, i.e. A_1. To estimate time, we apply Fitts' Law, considering that:

1. The measures of D and of the width of the target W are, respectively, the length of the segment c_0c_1 and of the height h of the A_0 trapezoid. They both have the same value l, as a consequence of simple geometrical considerations on the shapes shown in figure 4. This property does not hold for hexagonal and octagonal menu: the ratio D/W for them has, respectively, the value of 0.67 and 0.83.

2. for intercept and slope, we use the values experimentally obtained by [7] for dragging tasks on tablet devices: $a = -27; b = 276$.

Thus, by applying Fitts' Law, $MT(x_1) = a + b * lg2(2) = a + b$. To analyze the time for movements following the first one, $MT(x_i)$; $2 \leq i \leq n-2$, we have to analyze three possible cases:

a) movement between contiguous menu items;

b) movement between opposite menu items;

c) other movements (included the same menu item).

In case of type a) movements (i.e. between A_1 and A_4), D and W are, respectively, the length of the segment c_1c_4 and of the segment p_0p_1. They have the same length, thus by applying Fitts' Law, $MT(x_i) = a + b * log_2(2) = a + b$.

In case of type b) movements (i.e. between A_1 and A_3), D and W are, respectively, the length of the segment c_1c_3 and of the height of the A_3 trapezoid. D is twice W, thus by applying Fitts' Law, $MT(x_i) = a + b * log_2(3)$.

In case of type c) movements, the movement is regarded as composed of two simpler movements: a return in the key area followed by a new item selection, thus by applying Fitts' Law, $MT(x_i) = 2(a + b * log_2(2)) = 2(a + b)$.

Finally, the last movement can be a further character selection or a space. In the former case, $MT(x_{n-1})$ can be calculated as for the previous case. In the latter case, it is a return in the key area. It can be calculated by applying Fitts' Law, $MT(x_{n-1}) = a + b * log_2(2) = a + b$;

The time for menu scan can be calculated by applying the Hick-Hyman Law shown in (2). In particular, we use the following values for parameters:

- for intercept and slope, we use the values reported by [15]: $a' = 0; b' = 200$.

- for n, we use the number of menu items.

To calculate $RT(tu)$, we should add the above value for all the movements between menu items, except for movements for entering characters already selected in the same gesture. The above analysis, performed for movements inside a four items menu, is also valid for menus with more items.

4.2 Simulation

A software simulator has been developed, in order to calculate the time required to enter the text contained in an input text file. The simulator also accepts as input an XML file containing the definition of the keyboard layout. The output is a table which reports the time (in words per minutes) required to enter the text for each keyboard layout, both in its basic version and in the menu-augmented one.

Briefly, the simulator is equipped with a text scanner and a *TimeCalculator* module. The former scans the input text file and passes text units (single characters or strings matching using the regular expression (3)) to the latter. The *TimeCalculator* applies the rules described above in this section to calculate the time required to enter the text units.

4.3 Interpretation of results

The final result is a comparison of the performances on QWERTY layout. Menus with both single selection and multiple selection enabled with 4, 6, and 8 keys have been tested through the simulator. We have calculated the time required to enter the text contained in a corpus of Italian text, 58 books and journal articles published between years 1949 and 1996, freely downloadable from *Biblioteca della Scuola Normale* website [1]. The process of assigning the characters to the menu items has been aimed at minimizing movement times. In particular, from the calculations above, we know that type a) movements are faster than the other types. Thus, an optimal character arrangement should maximize movements between adjacent items.

Table 1 shows the results obtained by running the simulator with the text corpus on the QWERTY layout augmented with menu containing 4, 6 and 8 items. The second column in the table reports the sequence of the characters assigned to menu items, starting from the upper menu item and proceeding clockwise. The chosen letters are the most frequent in the Italian language. There is no warrant that this is the best choice for menu-augmented keyboards.

The arrangement has been performed trying to maximize movements between adjacent menu items. The following columns report the text entry speed in wpm: the third column reports the performances of the keyboard without any menu; the fourth column reports the performances of novice and expert users with a classical menu-augmented keyboard (one menu-item selection per time); lastly, the fifth column reports the performances of novice and expert users, with a multiple-selection enabled menu-augmented keyboard.

The results of our theoretical study are the following: for those users familiar with QWERTY layout, menu-augmented keyboards enable faster text entry speed when users become familiar with the menu layout too. Nevertheless, before acquainting themselves with it, their perfor-

Num of Items	char arrangement	QWERTY	Single Sel. Menu	Multiple Sel. Menu
4	aeoi	35.41	30.11 → 42.13	30.16 → 43.43
6	aenoli	35.41	26.87 → 41.58	25.21 → 43.30
8	aetnolri	35.41	24.78 → 40.97	20.35 → 38.94

Table 1. Performances (in words per minute) of menu augmented keyboards with QWERTY layout

mances are consistently lower. Our approach shows slightly better performances than classical menu-augmented keyboards with trained users with small menus (4 and 6 menu items). In particular, among our simulations, the fastest text entry has been obtained with the 4 items multiple-selection menu (43.32 wpm, with an improvement of 22.4% compared to simple QWERTY). Nevertheless, this result is not better than those obtained by other authors with optimized keyboard layout, such as *Fitaly* and *Opti II*.

Lastly, a result that could surprise the reader is that smaller menus have better performances than larger ones. We argue that this depends on the choice of the characters assigned to menu items. Other factors than character frequency can influence the performances of the method, such as the frequency of matches of the (3), and the keyboard layout: the character assigned to peripheral keyboard keys should be best candidates for the assignment to menu items, since their average distance from other keys is higher. Thus, their presence in the menu can help minimizing the pointer movements.

5. Conclusions and further research

We have presented a new text entry approach based on soft keyboards. The approach is an improvement of menu-augmented keyboards, consisting in the possibility of performing multiple selection of its items with a single pointer stroke. The approach has been theoretically evaluated in terms of text entry speed with an Italian text corpus. According to our simulation, expert users can input text faster than with classical QWERTY keyboard and also with QWERTY augmented with classical single selection menus, under certain circumstances. The above results encourage us to plan an experiment with human users, in order to measure the performances in real situations, from both the points of view of speed and accuracy. Before performing the experiment, further studies will be aimed at tuning our approach, by associating different characters to menu items and different menu instances to single keys.

References

[1] Biblioteca della 'scuola normale': Dizionari e corpora linguistici (in italian). http://biblio.sns.it/.

[2] C. H. Blickenstorfer. Graffiti: Wow!!!! *Pen Computing Magazine*, pages 30–31, 1995.

[3] P. M. Fitts. The information capacity of the human motor system in controlling the amplitude of movement. 1954. *J. of Exp. Psychology General*, 121(3):262–269, 1992.

[4] R. Hyman. Stimulus information as a determinant of reaction time. *J. of Exp. Psychology*, 45:188–196, 1953.

[5] P. Isokoski. Performance of menu-augmented soft keyboards. In *Proc. of CHI '04*, pages 423–430, New York, NY, USA, 2004. ACM.

[6] P.-O. Kristensson and S. Zhai. Shark2: a large vocabulary shorthand writing system for pen-based computers. In *Proc. of UIST '04*, pages 43–52, NY, USA, 2004. ACM.

[7] I. MacKenzie, A. Sellen, and W. Buxton. A comparison of input devices for elemental pointing and dragging tasks. In *Proc. of CHI 91*, pages 161–166. New York, 1991.

[8] I. S. MacKenzie. A note on the information-theoretic basis for fitts' law. *J. of Motor Behavior*, 21:323–330, 1989.

[9] I. S. MacKenzie and R. W. Soukoreff. Text entry for mobile computing: Models and methods, theory and practice. *Human-Computer Interaction*, 17:147–198., 2002.

[10] I. S. MacKenzie and S. X. Zhang. The design and evaluation of a high-performance soft keyboard. In *Proc. of CHI '99*, pages 25–31, New York, NY, USA, 1999. ACM.

[11] L. Magnien, J. L. Bouraoui, and N. Vigouroux. Mobile devices: soft keyboard text-entry enhanced by visual cues. In *Proc. of UbiMob '04*, pages 158–165, NY, USA, 2004. ACM.

[12] J. Mankoff and G. D. Abowd. Cirrin: a word-level unistroke keyboard for pen input. In *Proc. of the ACM UIST 98*, pages 213 – 214. ACM, 1998.

[13] The Open Group. *Regular Expressions*, the single unix specification, version 2 edition, 1997.

[14] K. Perlin. Quikwriting: continuous stylus-based text entry. In *Proc. of the ACM UIST 98*, pages 215 – 216. ACM, 1998.

[15] R. W. Soukoreff and I. S. MacKenzie. Theoretical upper and lower bounds on typing speed using a stylus and soft keyboard. *Behaviour and I.T.*, 14(6):370 – 379, 1995.

[16] Textware solutions: The fitaly one-finger keyboard. 1998. http://fitaly.com/fitaly/fitaly.htm.

[17] D. Venolia and F. Neiberg. T-cube: a fast, self-disclosing pen-based alphabet. In *Proc. of CHI '94*, pages 265–270, New York, NY, USA, 1994. ACM.

[18] S. Zhai, M. Hunter, and B. A. Smith. The metropolis keyboard - an exploration of quantitative techniques for virtual keyboard design. In *Proc. of UIST '00*, pages 119–128, New York, NY, USA, 2000. ACM.

[19] S. Zhai, M. Hunter, and B. A. Smith. Performance optimization of virtual keyboards. *Human-Computer Interaction*, 17:229–270, 2002.

[20] S. Zhai and P.-O. Kristensson. Shorthand writing on stylus keyboard. In *Proc. of CHI '03*, pages 97–104, New York, NY, USA, 2003. ACM.