2016 IEEE 28th International Conference on Tools with Artificial Intelligence

Modifying Keyboard Layout to Reduce Finger-travel Distance

Nan Yang
Department of Electrical Engineering & Computer Science
University of Wisconsin-Milwaukee
Milwaukee, WI 53211, USA
yn199145@gmail.com

Amol D. Mali
Department of Electrical Engineering & Computer Science
University of Wisconsin-Milwaukee
Milwaukee, WI 53211, USA
mali@uwm.edu

***Abstract*—Ample evidence shows that the Qwerty layout is inefficient for one-finger typing on virtual keyboards: letters in common digrams are placed on opposite sides of the keyboard, resulting in an unnecessarily long finger-travel distance. In this paper, we report on using simulated annealing for modifying keyboard layout to reduce finger-travel distance for entering letters of the English alphabet. Our investigation led to improved 3 X 10, 4 X 7, and 5 X 6 layouts, all of which reduced the weighted sum of finger-travel distances for all digrams (denoted by d) by about 40% over Qwerty. The most improved layout is the 5 X 6 layout, which reduces d from 3.31 key widths (the value of d for Qwerty) to 1.78 key widths. These layouts also outperform the best layout reported in the literature.**

Keywords*—Virtual Keyboard, Finger distance, Intelligent user interfaces

I. Introduction

Text entry on a mobile device via a virtual keyboard is one of the most common activities in the mobile computing era. ExactTarget survey (www.marketingcharts.com) shows that the top activities on smartphones are accessing e-mails and text messaging. Both involve a lot of text entry. A study (techlandtime.com) found that cell phone users between the ages of 18 and 24 years exchange 110 text messages per day on an average. Given the huge popularity of mobile text entry, any improvement in its performance would offer a huge benefit to a large number of users.

Qwerty has been the default layout for entering text for over a hundred years. However, it has been widely known to be inefficient for text entry on mobile devices. One of the rationales behind the Qwerty layout is that placing letters in common digrams on two sides of the keyboard requires a user to alternate between two hands during typing, and can avoid mechanical jams in typewriters. Such a layout turns out to be suboptimal for virtual keyboards on mobile devices, where many users enter text with only one finger. Because letters in common digrams are placed on two sides, the finger entering the text usually needs to travel back and forth multiple times between two sides of a keyboard, to enter a word. For example, Figure 1 shows the finger-travel trace for entering "open". As shown, the input finger travels across the keyboard twice for entering this simple four-letter word.

Long finger-travel distance is detrimental to the text entry experience. First of all, long finger-travel distance results in low input speed. Secondly, in one-handed typing scenario in which a user holds a phone with one hand and types using the thumb on the same hand, moving the thumb back and forth between two sides of a keyboard is unpleasant due to the limited range of thumb movement. Third, recent research shows that spending too much time texting can lead to neck and shoulder pain [1].

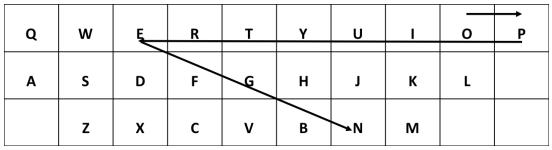

Fig. 1. Entering "open" on the Qwerty layout

Given the widely known disadvantages of the Qwerty layout, can we design a layout that is more efficient for one-handed typing? In this paper, we use artificial intelligence to investigate this question. We modify the keyboard layout in order to lower the weighted sum of finger-travel distances for all digrams. Our investigation shows that Qwerty indeed is a suboptimal layout for typing. d for this layout is 3.31 key widths. Applying simulated annealing, we obtained 3 X 10, 5 X 6, and 4 X 7 layouts for which d is lower by about 40% compared to Qwerty, substantially improving the efficiency of text entry.

II. Related Work

It has long been known that Qwerty is suboptimal for mobile text entry. As has been published [2], [3], Qwerty was initially designed to reduce jamming in mechanical typewriters by placing letters in common digrams on opposite sides of the keyboard. Such a layout is inefficient for one-finger text entry.

Researchers have explored different approaches to search for a better layout. Dunlop and Levine [4] applied a multidimensional Pareto optimization method to optimize the layout for three objectives: speed, familiarity with Qwerty, and improved spell-checking ability. Their focus was on 3 X 10 layout, while we removed the constraint on the number of rows and columns and improved 5 X 6 and 4 X 7 layouts as well. Dunlop and Levine had three objectives in their procedure, while we focused on shortening finger-travel distance. Oulasvirta et al. [5] proposed the KALQ layout, for efficient two-thumb typing. Such a layout was designed to reduce thumb-travel distance and increase alternation between

2375-0197/16 $31.00 © 2016 IEEE
DOI 10.1109/ICTAI.2016.31

thumbs. Bi et al. [6] proposed Quasi-Qwerty layout, which was designed to improve input speed, and maintain a certain level of similarity with Qwerty.

In addition to improving the layout for regular typing, there has been a sizable amount of research improving the keyboard layout for gesture typing, a text entry method that allows a user to enter a word by gliding the finger from one letter to another. Smith et al. [7] have proposed a layout for improved gesture input speed, gesture recognition accuracy, and similarity with Qwerty. Bi and Zhai [8] proposed the IJQwerty layout on which positions of I and J were swapped. Such a layout improves the performance of gesture typing over Qwerty. Prior work focused on 3 X 10 layout. Our work focuses on reducing d exclusively, and explores the search space for 3 X 10, 4 X 7 and 5 X 6 layouts.

III. Research Goals

Our primary goal is to reduce d on a virtual keyboard by rearranging the keys. Lower finger-travel distance will lead to higher input speed and less physical movement of finger. We investigate improvement of the keyboard for 5 X 6 and 4 X 7 layouts, because of the following two reasons. First of all, square-like layouts have a lot of potential for reducing finger-travel distance because the maximum distance between two keys considering all pairs of keys is reduced compared to a layout with less rows. Second, square-like layouts suit one-handed typing better than a Qwerty-like 3 X 10 layout.

IV. Improvement Procedure

The objective function of our improvement procedure is d, computed by the following equation.

$$d = \sum_{i=1}^{26} \sum_{j=1}^{26} P_{ij} D_{ij} \quad (1)$$

D_{ij} is the distance between key i and key j on the layout, and P_{ij} is the probability of occurrence of digram ij. d is the weighted sum of finger-travel distances for all digrams. The distance between keys i and j is the Euclidean distance between the bottom-left corners of these keys. P_{ij} is found by dividing the total number of occurrences of digram ij in all words in the chosen collection of words or sentences by the total number of occurrences of all digrams in the same collection of words or sentences.

We used simulated annealing to improve keyboard layouts, because of the success of simulated annealing in solving large combinatorial instances. Simulated annealing is inspired by annealing in metallurgy. Simulated annealing in metallurgy is heating and controlled cooling of a material to increase the size of its crystals and reduce their defects. Unlike purely greedy algorithms, simulated annealing occasionally accepts worse options. On each iteration, Algorithm 1 randomly swaps two keys on the layout. If d for the new keyboard layout (calculated using Equation (1)) is lower than or equal to that for the old one, the new layout is kept and the algorithm advances to the next iteration. If the new layout is worse than the old layout, the new layout is accepted with a probability found using Metropolis function, shown in Equation (2).

$$W(O-N) = \begin{cases} e^{-\Delta d/T}, & \text{if} \quad \Delta d > 0. \\ 1, & \text{if} \quad \Delta d \leq 0. \end{cases} \quad (2)$$

$W(O-N)$ is the probability of changing from layout O (old) (this is same as the current layout) to layout N (new); $\Delta d = d_{new} - d_{old}$, where d_{new} and d_{old} are found using Equation (1) and T is the "temperature" in simulated annealing. Figure 2 shows the computational flow of our algorithm. The value of α is chosen by the user. Function $swap(i, j, k)$ returns the keyboard layout obtained from keyboard layout k by swapping the positions of keys i and j. $getMetropolisValue$ function computes $metropolisValue$ using $distanceNew$, $distanceOld$, and T.

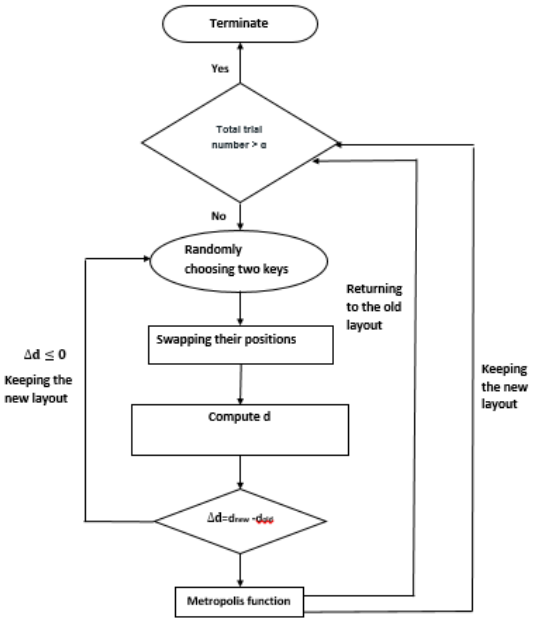

Fig. 2. Computational Flow of the Algorithm

V. Experimental Results

We implemented our keyboard-improvement algorithm in JAVA, on a laptop with Windows 10 Home, 64-bit Operating System, 8 GB RAM, and 2048 KB ROM. The processor is Intel Core i5-4200M CPU @2.5 GHz. We improved keyboard layouts using American National Corpus (ANC) (www.anc.org), which is an open-source English corpus with over 233,000 words. We took all ANC words into consideration in computing P_{ij} for all digrams.

Figures 3, 4 and 5 show the improved keyboards for 3 X 10, 4 X 7 and 5 X 6 layouts respectively. Each layout was obtained after 100,000 trials (the value of α). Our computer was able to complete these trials in 20 cpu seconds for each layout. Cartesian coordinates were assigned to the bottom-left corner of each key to measure the distance between keys, e.g. in Fig. 3, the coordinates of E are (0,0), the coordinates of G are (-3,-1), and the coordinates of Q are (5,1). In Fig. 4, the coordinates of A are (0,0), the coordinates of B are (3,2), and the coordinates of J are (-2,-1).

We obtained the improved layouts shown in figures 3, 4, and 5, by starting from a random layout. To get better results, we ran Algorithm 1 for 5 X 6 layout again, with a different

Algorithm 1

procedure IMPROVE-KEYBOARD(*currentKeyboard*)
  totalTrials $\leftarrow 0$
  bestLayout $\leftarrow$ *currentKeyboard*
  shortestDistance $\leftarrow distance($*bestLayout*$)$
  keyboardOld $\leftarrow$ *currentKeyboard*
  distanceOld $\leftarrow distance(keyboardOld)$
  while *totalTrials* $\leq \alpha$ **do**
    key $i \leftarrow randomly\ chosen\ key$
    key $j \leftarrow$ *randomly chosen key different from* i
    keyboardNew $\leftarrow$ *swap(i,j,keyboardOld)*
    distanceNew $\leftarrow distance($*keyboardNew*$)$
    if *distanceNew* $\leq$ *shortestDistance* **then**
      bestLayout $\leftarrow$ *keyboardNew*
      shortestDistance $\leftarrow$ *distanceNew*
    if *distanceNew* $\leq$ *distanceOld* **then**
      keyboardOld $\leftarrow$ *keyboardNew*
      distanceOld $\leftarrow$ *distanceNew*
    else if *distanceNew* $>$ *distanceOld* **then**
      metropolisValue $\leftarrow$ *getMetropolisValue()*
      randomValue $\leftarrow$ *a random number from* $[0, 1]$
      if *randomValue* $\leq$ *metropolisValue* **then**
        keyboardOld $\leftarrow$ *keyboardNew*
        distanceOld $\leftarrow$ *distanceNew*
    totalTrials++
  return *bestLayout*

Z	W	H	T	S	O	U	F	J	Q
K	V	C	I	E	R	M	B	X	
	G	D	N	A	L	P	Y		

Fig. 3. Improved 3 X 10 layout. d for this is 1.93 key widths.

Z	K	W	H	C	M	B
X	V	I	T	E	R	Y
Q	G	N	A	S	O	F
	J	D	L	U	P	

Fig. 4. Improved 4 X 7 layout. d for this is 1.83 key widths.

initial arrangement of the keys. The letters in high-frequency digrams were put adjacent to each other in the initial layout. The initial layout is shown in Figure 6. d for this is 1.916 key widths. In this initial layout, we placed the letter with the highest frequency (i.e. "E") in the center of the keyboard. Then letters in high-frequency digrams containing E were placed

Z	V	B	D		
C	H	G	F	O	P
E	S	M	W	L	A
Y	R	N	T	X	Q
		I	U	K	J

Fig. 5. Improved 5 X 6 layout. d for this is 1.78 key widths.

V	W	C	Y		
M	N	I	S	G	X
L	O	E	T	D	Q
F	R	A	H	P	J
		U	B	K	Z

Fig. 6. The original 5 X 6 layout in which letters in frequent digrams are placed adjacent to each other. d for this is 1.916 key widths.

K	G	D	V		
F	N	I	H	W	Z
M	O	E	T	C	X
P	R	A	S	U	J
		L	Y	B	Q

Fig. 7. Another improved 5 X 6 layout. d for this is 1.795 key widths.

based on the position of E. According to our calculation, the top eight most frequent digrams including E are ER, EN, ES, EO, EA, ET, EH, and EI. We placed the letters in these eight digrams adjacent to E. Then we placed remaining letters from other high-frequency digrams which had the letters recently added to the keyboard. We repeated this until all the twenty-six letters were placed. This did not result in the best layout, but

this yielded a better starting layout than a randomly generated layout. With this as the starting point, we ran Algorithm 1 for 20,000 iterations. The algorithm led to a better layout (Figure 7), with d equal to 1.795 key widths, close to the value of d for the layout shown in Figure 5. Figure 8 shows the best value of d we obtained for each of the layouts, along with the values of d for Qwerty and the layout in [4] that we found using Equation (1). As shown, the least value of d in key widths that we obtained is 1.93 for 3 X 10 layout, 1.83 for 4 X 7 layout, and 1. 78 for 5 X 6 layout. All of these values are at least 40% lower than that for Qwerty (3.31 key widths). The values of d we obtained also outperformed the layout proposed in [4].

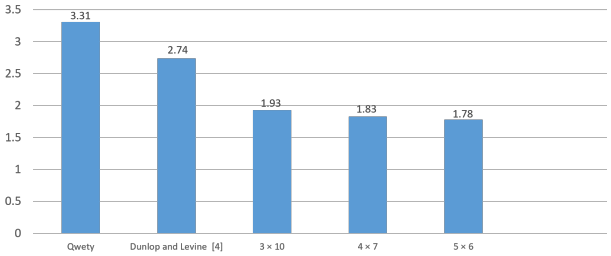

Fig. 8. d for different layouts, found with Equation (1)

VI. Conclusion

We used iterative search to improve keyboard layout to improve text-input speed and reduce the physical effort spent on entering text. We focused on reducing the weighted sum of finger-travel distances for all digrams. We investigated 4 X 7 and 5 X 6 layouts, besides improving the existing 3 X 10 layout. We were able to find highly improved versions of all three layouts which are also superior to Qwerty. The weighted sum of finger-travel distances for all digrams for each of our layouts is over 40% lower than that for Qwerty layout. Our best layout is the 5 X 6 layout, for which the weighted sum of finger-travel distances for all digrams is only 1.78 key widths compared to 3.31 key widths for Qwerty. The improved versions of all three layouts we investigated also outperformed the best layout reported in the literature, for which the weighted sum of finger-travel distances for all digrams equals 2.74 key widths.

Acknowledgment

We thank Ethan Munson for his comments and suggestions.

References

[1] "Too much texting linked to neck and shoulder pain," http://www.medicalnewstoday.com/articles/170953.php.

[2] J. Rick "Performance optimizations of virtual keyboards for stroke-based text entry on a touch-based tabletop," *Proceedings of the 23rd Annual ACM Symposium on User Interface Software and Technology*, 2010, pp. 77-86.

[3] H. Yamada, "A historical study of typewriters and typing methods, from the position of planning Japanese parallels," *Journal of Information Processing*, 2, 1980, pp. 175-202.

[4] M. Dunlop and J. Levine, "Multidimensional pareto optimization of touchscreen keyboards for speed, familiarity and improved spell checking," *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2012, pp. 2669-2678.

[5] A. Oulasvirta, A. Reichel, W. Li, Y. Zhang, M. Bachynskyi, K. Vertanen, and P. O. Kristensson, "Improving two-thumb text entry on touchscreen devices," *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2013, pp. 2765-2774.

[6] X. Bi, B. Smith, and S. Zhai, "Quasi-qwerty soft keyboard optimization," *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2010, pp. 283-286.

[7] B. Smith, X. Bi, and S. Zhai, "Optimizing touchscreen keyboards for gesture typing," *Proceedings of the 33rd Annual ACM SIGCHI Conference on Human Factors in Computing Systems*, 2015, pp. 3365-3374.

[8] X. Bi and S. Zhai, "IJqwerty: What difference does one key change make? Gesture typing keyboard optimization bounded by one key position change from Qwerty," *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2016, pp. 49-8.