Modeling Text Input for Single-Switch Scanning

I. Scott MacKenzie

Dept. of Computer Science and Engineering,
York Univeristy,
Toronto Ontario Canada M3J 1P3
mack@cse.yorku.ca

Abstract. A method and algorithm for modeling single-switch scanning for text input is presented. The algorithm uses the layout of a scanning keyboard and a corpus in the form of a word-frequency list to generate codes representing the scan steps for entering words. Scan steps per character (*SPC*) is computed as a weighted average over the entire corpus. *SPC* is an absolute measure, thus facilitating comparisons of keyboards. It is revealed that *SPC* is sensitive to the corpus if a keyboard includes word prediction. A recommendation for other research using *SPC* is to disclose both the algorithm and the corpus.

Keywords: Single-switch scanning, text input, models of interaction, scan steps per character.

1 Introduction

One enduring method of accessible text entry is single-switch scanning using a virtual keyboard. Keys are highlighted in sequence ("scanned") with selections made with a single input switch when the key bearing the desired character is highlighted. To speed-up entry, a two-tier scanning pattern is often used, such as row-column scanning. Scanning thus proceeds row by row. When the row bearing the desired character is highlighted, it is selected. Scanning proceeds within the row until the key bearing the desired character is highlighted. A selection adds the character to the text message, with scanning restarted at the top row.

To further speed-up entry, the keyboard is often arranged with high-frequency letters (e.g., *e*, *a*, or *t*, for English) near the beginning of the scanning sequence. This reduces the required number of scan steps to reach letters. To speed-up entry even further, word prediction techniques are often added. This allows a word to be selected after only part of the word is entered.

It is apparent from the points above that text entry using single-switch scanning can be modeled analytically. The rate of entry will depend on the scanning interval and the number of scan steps to enter letters or words. The physical demand on users will depend on the required number of switch activations to enter letters or words. Techniques to optimize aim to lower the number of scan steps or the number of switch activations. Comparing the computed numbers can assess alternate designs.

This paper is on modeling text entry using single-switch scanning. The goal is to both review existing modeling techniques in the literature and to suggest improvements to these techniques. The most serious deficiency in existing models is the use

K. Miesenberger et al. (Eds.): ICCHP 2012, Part II, LNCS 7383, pp. 423–430, 2012.

of relative metrics. Relative metrics reveal the percent drop in scan steps when a design is optimized in some way. While relevant for variations on a design, relative metrics cannot be used to compare designs across publications, because the baseline design is different, not precisely known, or inherently dissimilar. Because of this, the body of research on text entry using single-switch scanning is a collection of distinct publications that cannot be compared. The work does not form a unified coherent body of research. One goal here is to correct this.

2 Relative versus Absolute Metrics

As noted above, the efficiency of text entry using single-switch scanning can be improved using a variety of techniques. Research on this spans several decades [1-7], [13-16]. The problem alluded to above is apparent in Lesher et al.'s review [7] where comparisons of alternate designs are expressed in relative terms. As an example, row-column scanning using an alphabetic letter arrangement is contrasted with an optimized letter-arrangement. See Fig.1.a and Fig.1.b. It is noted that the optimized arrangement is "more efficient" [7, p. 83], but no statistics are given. For other comparisons, statistics are given, but only in relative terms, for example, "incorporation of a word list provides a 24% savings in switch counts" [7, p. 84]. (Note: a "switch count", as used here, is the same as a "scan step"). The full details of the baseline design are not given, nor are any absolute measures provided. The alluded-to 24% savings is meaningful only within the cited publication. In another paper, we learn of "keystroke savings … in the range of 37-47%" [4]; but, again, the baseline design is insufficiently detailed and no absolute measures are given.

The examples above are not unique. Despite a considerable body of work, it is difficult, arguably impossible, to compare designs from one publication to the next. Mankowski echoes this sentiment: "unfortunately, they only report the relative savings, not the actual number of scan steps" [11, p, 103].

This limitation is easily rectified, drawing upon text entry modeling from mobile computing. The efficiency of text entry on phone keypads and their variants is aptly reflected in keystrokes per character (*KSPC*) [8], [10]. *KSPC* is the average number of keystrokes required to produce each character of text on a given keyboard, in a given language, using a given entry method. The statistic is an absolute measure. The equivalent for scanning keyboards is scan-steps per character (*SPC*). *SPC* is calculated a similar way except using the average number of scanning intervals to produce a character of text.[1] *SPC* is computed as a weighted average for a language and requires a letter-frequency list or word-frequency list derived from a corpus.

To illustrate the utility of *SPC*, Fig.1. shows 12 scanning keyboards from 7 sources. For this analysis, only the core letters (*a*-*z*) and *space* (_) are shown. Letters are in lowercase. Where word prediction is used, *Word* appears. Additional columns on the right or rows at the bottom, if present, are not shown, as these do not impact the calculation of *SPC* (assuming top-to-bottom, left-to-right scanning).

[1] A related statistic is selections per scan-step, *SPS*, which reflects the motor demand of entry. *SPS* is not elaborated in this paper.

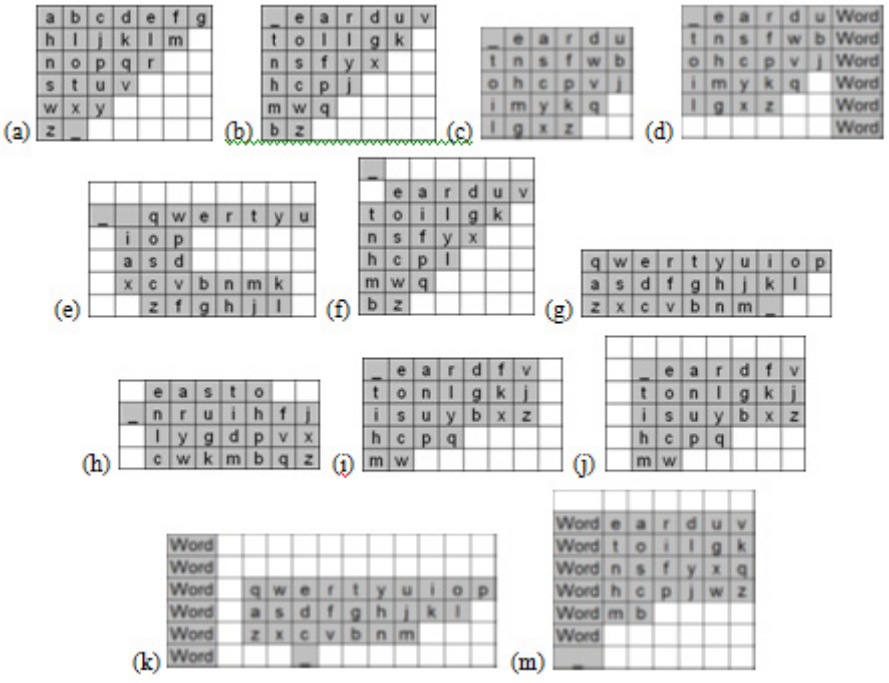

Fig. 1. Twelve scanning keyboards. See Fig. 2 for further details.

The layouts in Fig.1. are diverse, suggesting vastly different objectives in each design. Scan-steps per character (SPC) is useful because it enables comparing designs from different sources.

3 Coding Examples and Calculation of *SPC*

To calculate SPC, the keyboard geometry and operation are embedded in an algorithm implemented in software. A corpus in the form of a word-frequency list is also required. The algorithm processes each word to determine the scan steps required to enter the word. Some scan steps are passive (no selection) while others require a switch activation. A space is assumed to follow each word. Creating a graphical representation of the scan steps is important both for debugging and for examining and demonstrating the keyboard's operation. See Fig.2. For this demonstration, the optimized keyboard by Koester and Levine [6] was used (see Fig. 1c and 1d). Three variations are included along with scan steps for some example words. The words are the, of, and, weapons, spot, and slow. The word rankings are from the BNC-1 corpus (discussed below). The scan steps include the entry of a terminating space.

Passive scan steps are coded as periods. The methods in Fig.2a and Fig.2b use conventional row-column scanning. A row selection appears as an uppercase R. Switch activations generating characters appear as the corresponding lowercase letter. For word prediction, a selection in the word list appears as uppercase W.

	Keyboard	Rank	Word	Freq.	Scan steps	Notes
(a)	Fig. 1c	1	the	5,776,384	.Rt..R.hR.eR_	Optimized (row-column scanning)
		2	of	2,789,403	..Ro.R...fR_	
		3	and	2,421,302	R..a.R.nR....dR_	
		2000	weapons	3,861	.R....wR.eR..a..R...p..Ro.R.n.R..sR_	
		2001	spot	3,859	.R..s..R...p..Ro.RtR_	
		2002	slow	3,855	.R..s....Rl..Ro.R....wR_	
(b)	Fig. 1d	1	the	5,776,384	R......W	Optimized + word prediction (row-column scanning)
		2	of	2,789,403	.R......W	
		3	and	2,421,302	..R......W	
		2000	weapons	3,861	R......wR.eR..a..R......W	
		2001	spot	3,859	.R..s..R...p..Ro...R......W	
		2002	slow	3,855	.R..s....Rl..R......W	
(c)	Fig. 1d	1	the	5,776,384	.HW	Optimized + word prediction (half & half scanning)
		2	of	2,789,403	.H.W	
		3	and	2,421,302	.H..W	
		2000	weapons	3,861	HR......wHR.eHR..a.H..W	
		2001	spot	3,859	H.R..sH..R...pH..Ro.H...W	
		2002	slow	3,855	H.R..sH....Rl.H..W	

Fig. 2. Coding examples. See text for discussion.

The method in Fig.2.c uses "half-and-half scanning" [6, p. 44]. To speed-up entry with word prediction, scanning initially alternates between the letter-half (left) and the word-half (right) of the keyboard (see Fig.1.d). The user first selects the desired half (*H* in Fig.2.c), then selects within the letter or word half of the keyboard. Within the letter region, conventional row-column scanning is used.

Word prediction allows a word to be selected before all letters are entered. The algorithm assumes the word is selected at the earliest opportunity. This will depend on the corpus and the size of the word list. The keyboard in Fig. 1d includes a word list with six entries. The list appears before the first letter of a word is entered and is updated during entry according to the current word stem. At the beginning of a word, the list is simply the six most-frequent words in the corpus. These words can be entered by selecting the word half of the keyboard (*.H*), then selecting the word, for example, *W* for *the*, *.W* for *of*, or *..W* for *and*. See Fig.2.c.

Less common words must be partially or fully entered before word selection is possible. As seen in the last row in Fig.2., *slow* was entered after the input of *sl*. After *s*, the list contained *she*, *said*, *some*, *so*, *should*, and *such* – the six most frequent words beginning with *s*. After *sl*, the list contained *slightly*, *slowly*, *slow*, *sleep*, *slight,* and *slipped. Slow* is the third entry. The user selects the word half of the keyboard (*.H*), then selects *slow* (*..W*). See Fig. 4c, last row.

With software to generate the scan step codes, as in Fig.2., scan steps per character (*SPC*) is calculated as a weighted average over the entire corpus:

$$SPC = \frac{\sum (S_W \times F_W)}{\sum (C_W \times F_W)} \tag{1}$$

where SW is the number of scan steps to enter a word, FW is the frequency of the word, and CW is the number of characters in the word. SW and CW are adjusted to include a terminating space after each word.

The *SPCs* for the 12 layouts in Fig.1. are given in Fig.3. Values are given for four corpora (discussed below). The values range from a low of *SPC* = 2.45 for Fig.1.m to a high of *SPC* = 7.45 for Fig.1.g.

Fig. 1	Source (1st author, year, reference)	*SPC* (by Corpus)				Spread (%)	Notes
		BNC-1	BNC-2	Brown	Phrases		
a	Lesher, 1998, [7, Fig. 1b]	5.78	5.79	5.77	5.88	1.9%	alphabetic
b	Lesher, 1998, [7, Fig. 1c]	4.27	4.30	4.31	4.34	1.6%	optimized
c	Koester, 1994, [6, Fig. 2]	4.28	4.32	4.32	4.36	1.8%	optimized
d	Koester, 1994, [6, Fig. 3]	3.35	3.44	4.34	3.20	26.3%	optimized + word pred. (row/col scanning)
d	Koester, 1994, [6, Fig. 3]	2.73	2.87	2.89	2.55	11.8%	optimized + word pred. (half & half scanning)
e	Steriadis, 2003, [14, Fig. 29]	7.40	7.42	7.44	7.38	0.8%	qwerty
f	Steriadis, 2003, [14, Fig. 30]	5.08	5.12	5.13	5.16	1.6%	optimized
g	Bhattacharya, 2008, [1, Fig. 1]	7.43	7.41	7.40	7.45	0.7%	qwerty
h	Szeto, 1993, [15, Fig. 1]	5.35	5.36	5.37	5.37	0.4%	optimized
i	Mankowski, 2009, [11, p. 105]	4.28	4.31	4.32	4.34	1.4%	optimized
j	Mankowski, 2009, [11, p. 105]	6.28	6.31	6.32	6.34	0.9%	optimized with row/col delay
k	http://wivik.com/	3.80	4.08	4.17	3.49	16.3%	qwerty + word pred.
m	http://wivik.com/	2.64	2.80	2.84	2.45	13.7%	optimized + word pred.

Fig. 3. Scan-steps per character (*SPC*) for the 12 keyboards in Fig.1., computed using the BNC-1, BNC-2, Brown, and Phrases corpora. See text for discussion

The benefit of using an absolute metric like SPC to model scanning keyboards is apparent in Fig.3.. Using SPC, comparisons are possible both for designs from the same source and for designs from different sources. For example, the first two rows in Fig.3. are for the alphabetic and optimized layouts in Lesher et al. [7]. The optimized layout is more efficient since it can produce text with about $(5.78 - 4.27) / 5.78 \times 100 = 26.1\%$ fewer scan steps per character. Although the difference is expressed as a percentage, the calculation requires SPC, an absolute metric. Provided SPC is reported, comparisons are possible for keyboards in different publications. For example, consider the Koester-Levine keyboard in Fig.1.d and the Wivik keyboard in Fig.1.m. Both are optimized keyboards using word prediction. The Koester-Levine keyboard places the word list on the right but uses half-and-half scanning to speed-up entry. The Wivik keyboard places the word list on the left and uses conventional row-column scanning. Evidently, the Wivik keyboard is slightly more efficient, since it requires about $(2.73 - 2.64) / 2.73 \times 100 = 3.3\%$ fewer scan steps per character of text. Although given in relative terms, the comparison is only possible because of the availability of SPC, an absolute metric.

4 Corpus Effect

The calculation of SPC is sensitive to the linguistic structure of the corpus. To test for a possible "corpus effect", SPC values were computed for four corpora: the British

National Corpus (BNC-1, BNC-2) [12], the Brown Corpus (see Wikipedia), and Phrases [9]. Each corpus was reduced to a word-frequency list for computing SPC. The Phrases list was built from a phrase set commonly used to evaluate text entry methods. It is only included here to test the effect of using a very small corpus on the calculation of SPC. The linguistic structure of the four corpora is given in Fig.4.

Corpus	Unique Words	Total Words
BNC-1	9,022	67,962,112
BNC-2	64,588	90,563,847
Brown	41,532	997,552
Phrases	1,163	2,712

Fig. 4. Characteristics of corpora used for calculations of *SPC* in Fig. 3

The spread of SPC values by corpus in Fig.3. is less than 2% for all designs not using word prediction. So, in the absence of word prediction, SPC is relatively insensitive to the corpus. However, the spread varies from 11.8% to 26.3% for the four designs using word prediction. In general, larger dictionaries yield higher SPC values. The wide spread for designs using word prediction is a problem. Comparisons of keyboards from different publications will be compromised, perhaps wrong, unless the SPC calculations use the same algorithm and the same corpus.

To illustrate the problem, let's revisit the comparison above – between the Koester-Levine keyboard (Fig.1.d) and the *Wivik* keyboard (Fig.1.m). The observation that the *Wivik* keyboard is about 3.3% more efficient was based on the *SPCs* calculated using the BNC-1 corpus. What if the comparison used *SPCs* calculated using a different corpus for each keyboard? This question is explored in Fig.5. which provides a cross-corpus comparison of *SPCs* for the two keyboards. The 3.3% figure just cited is in the top-left cell. Provided the *SPCs* are calculated from the same corpus (diagonal entries), the conclusion is consistent: The *Wivik* keyboard is slightly more efficient. However, if the comparison is based on *SPCs* calculated using a different corpus for each keyboard, the effect is dramatic. See Fig.5.. One comparison gives the *Wivik* keyboard a 15.2% advantage; another comparison gives it an 11.4% disadvantage! Clearly, research on scanning keyboards using *SPC* as an indicator of keyboard efficiency should disclose both the algorithm and the corpus.[2]

			SPCs for Koester-Levine keyboard (Fig. 1d)			
			BNC-1	BNC-2	Brown	Phrases
			2.73	2.87	2.89	2.55
SPCs for *Wivik* keyboard (Fig. 1m)	BNC-1	2.64	3.3%	8.0%	8.7%	-3.5%
	BNC-2	2.80	-2.6%	2.4%	3.1%	-9.8%
	Brown	2.84	-4.0%	1.0%	1.7%	-11.4%
	Phrases	2.45	10.3%	14.6%	15.2%	3.9%

Fig. 5. Cross-corpus comparison of *SPCs* for two scanning keyboards

[2] The software and word-frequency lists used to generate the *SPC* statistics herein are available at `http://www.yorku.ca/mack/ScanningKeyboardSPC.zip`

5 Discussion

If scan-steps per character (SPC) is calculated only using the core symbols (a-z, space), the result is robust but does not generalize to the broader text entry experience. To improve external validity, the model must include punctuation and other symbols. The first step is to position the added symbols in the layout. A revised corpus is also required – to include the added symbols. There is a problem, however. While common symbols, such as periods and commas, are relatively stable within corpora, many other symbols are not. Consider the variety of symbols on a conventional keyboard (e.g., ~, #, &, ^). Corpora that include these are likely to do so erratically. The symbols will be rare in some text samples, more frequent in others. SPC, so calculated, will address a broader context (high external validity), but will be unstable (low internal validity), thus weakening comparisons between designs from different sources.

The calculation of SPC does not factor in the human element. The statistic is purely a measure of inherent efficiency. Of course, assuming perfect behaviour for the operator (i.e., no errors, no selections missed), text entry throughput can be calculated from the scanning interval and SPC. Additional behaviours must be accounted for, however. Many scanning keyboards include a row delay, timer restart, or other adjustments to the scanning pattern to aid the user. Text entry throughput can be calculated if these properties are included along with SPC and the scanning interval. The calculation is a "best case". User efficiency can thereafter be measured as the ratio of the observed throughput to the best-case throughput.

6 Conclusion

Scan steps per character (SPC) was demonstrated as metric for modeling scanning keyboards. Since SPC is an absolute metric, designs from different sources can be compared. Where word prediction is used, comparisons are strengthened if SPC is calculated using the same corpus (word-frequency list).

References

1. Bhattacharya, S., Samanta, D., Basu, A.: Performance models for automatic evaluation of virtual scanning keyboards. IEEE Transactions on Neural Systems and Rehabilitation Engineering 16, 510–519 (2008)
2. Damper, R.I.: Text composition by the physically disabled: A rate prediction model for scanning input. Applied Ergonomics 15, 289–296 (1984)
3. Foulds, R., Baletsa, G., Crochetiere, W.: The effectiveness of language redundancy in nonverbal communication. In: Proceedings of the Conference on Devices and Systems for the Disabled, pp. 82–86. Krushen Center for Research and Engineering, Philadelphia (1975)
4. Higginbotham, D.J.: Evaluation of keystroke savings across five assistive communication technologies. Augmentative and Alternative Communications 8, 258–272 (1992)

5. Jones, P.E.: Virtual keyboard with scanning and augmented by prediction. In: Proceedings of the 2nd European Conference on Disability, Virtual Reality and Associated Technologies, pp. 45–51. University of Reading, UK (1998)
6. Koester, H.H., Levine, S.P.: Learning and performance of able-bodied individuals using scanning systems with and without word prediction. Assistive Technology 6, 42–53 (1994)
7. Lesher, G., Moulton, B., Higginbotham, D.J.: Techniques for augmenting scanning communication. Augmentative and Alternative Communication (AAC) 14, 81–101 (1998)
8. MacKenzie, I.S.: KSPC (Keystrokes per Character) as a Characteristic of Text Entry Techniques. In: Paternó, F. (ed.) Mobile HCI 2002. LNCS, vol. 2411, pp. 195–210. Springer, Heidelberg (2002)
9. MacKenzie, I.S., Soukoreff, R.W.: Phrase sets for evaluating text entry techniques. In: Extended Abstracts of the ACM SIGCHI Conference on Human Factors in Computing Systems - CHI 2003, pp. 754–755. ACM, New York (2003)
10. MacKenzie, I.S., Tanaka-Ishii, K.: Text entry with a small number of buttons. In: MacKenzie, I.S., Tanaka-Ishii, K. (eds.) Text Entry Systems: Mobility, Accessibility, Universality, pp. 105–121. Morgan Kaufmann (2007)
11. Mankowski, R.E.: Predicting communication rates: Efficacy of a scanning model, MSc Thesis, Univeristy of Pittsburg (2009)
12. Silfverberg, M., MacKenzie, I.S., Korhonen, P.: Predicting text entry speed on mobile phones. In: Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems, CHI 2000, pp. 9–16. ACM, New York (2000)
13. Simpson, R.C., Koester, H.H.: Adaptive one-switch row-column scanning. IEEE Transactions on Rehabilitation Engineering 7, 464–473 (1999)
14. Steriadis, C.E., Constantinou, P.: Designing human-computer interfaces for quadriplegic people. ACM Transactions on Computer-Human Interaction (TOCHI) 10, 87–118 (2003)
15. Szeto, A.Y.J., Allen, E.J., Littrell, M.C.: Comparison of speed and accuracy for selected electronic communication devices and input methods. Augmentative and Alternative Communication 9, 229 (1993)
16. Venkatagiri, H.S.: Efficient keyboard layouts for sequential access in augmentative and alternative communication. Augmentative and Alternative Communication (AAC) 15, 126–134 (1999)