The One-Key Challenge: Searching for a Fast One-Key Text Entry Method

I. Scott MacKenzie
Dept. of Computer Science and Engineering
York University
Toronto, Canada M3J 1P3
mack@cse.yorku.ca

ABSTRACT
A new one-key text entry method is presented. SAK, for "scanning ambiguous keyboard", combines one-key physical input (including error correction) with three virtual letter keys and a SPACE key. The virtual letter keys are highlighted in sequence ("scanned") and selected when the key bearing the desired letter receives focus. There is only one selection per letter. Selecting SPACE transfers scanning to a word-selection region, which presents a list of candidate words. A novel feature of SAK is multiple-letter-selection in a single scanning interval. In an evaluation with 12 participants, average entry speeds reached 5.11 wpm (all trials, 99% accuracy) or 7.03 wpm (error-free trials). A modification using "timer restart on selection" allowed for more time and more selections per scanning interval. One participant performed extended trials (5 blocks x 5 phrases/block) with the modification and reached an average entry speed of 9.28 wpm.

Categories and Subject Descriptors
H.5.2 [**Information Interfaces and Presentation**]: User Interfaces – *input devices and strategies (e.g., mouse, touchscreen)*; K.4.2 [**Computer and Society**]: Social Issues – *assistive technologies for persons with disabilities*

General Terms
Design, Experimentation, Human Factors

Keywords
Text entry, keyboards, ambiguous keyboards, scanning keyboards, mobile computing, assistive technologies.

1. INTRODUCTION
At a recent tutorial on text input, the presenter suggested that the ubiquitous mobile phone keypad could be redesigned with fewer keys. If text entry is possible using a keypad with 26 letters on 8 keys, why not put the letters on 7 keys, or 6 keys, or 5 keys, or 4 keys, or 3 keys, or 2 keys, or 1 key? See Figure 1. The audience laughed at the thought of a one-key keyboard. But, wait. The presenter noted that people with severe physical disabilities are often constrained to single-switch input to computers. The switch may be operated by the hand or foot, or perhaps even by a head movement, the blink of an eye, or sipping and puffing on a straw. Of course, the need and desire to use computer technology and to communicate, often through text, is the same for disabled people as it is for the able-bodied.

This paper lays out the one-key challenge for text entry. Is one-key text entry possible? How quickly and accurately can it be done? Below, we examine and review research on small keyboards. The review includes ambiguous keyboards, such as the phone keypad, and smaller versions of ambiguous keyboards. We also review scanning keyboards, as used by people with disabilities. A method is described to combine scanning with an ambiguous keyboard to arrive at a design where one-key text entry is not only possible, but is reasonably fast and accurate. The results of a user study are given with comparisons drawn to results from related user studies.

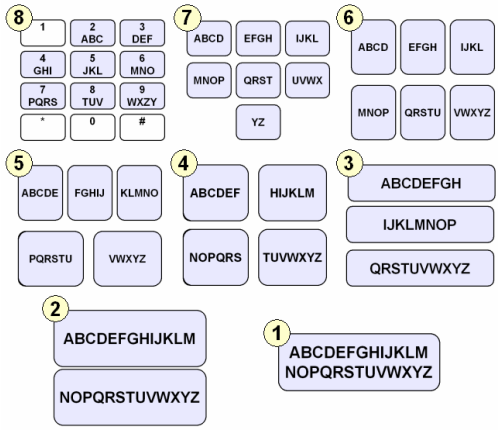


Figure 1. The one-key challenge: Design a fast text entry method using just a single physical key for input.

1.1 Mobile Phone Text Entry
According to gsmworld.com, one trillion or more text messages are sent yearly from one mobile phone to another. Most are entered using the conventional 12-key phone keypad, which positions the 26 letters of the English alphabet across just eight keys (Figure 1, top left). The arrangement is ambiguous for text entry, since each key bears either three or four letters. There are two common methods to disambiguate. "Multitap" involves

ASSETS'09, October 25–28, 2009, Pittsburgh, Pennsylvania, USA.

pressing a key multiple times: one press for the 1st letter, two for the 2nd, and so on. For example, "beep" is entered using multitap as

```
233n3370
b e  ep_
```

The marker "n", for "next", reveals the need to separate consecutive letters on the same key. Typically, a timeout or a special key such as down-arrow is used. A "0" is also included to add a SPACE at the end of a word. The other method is "one-key with disambiguation", commonly known as "T9". With T9, there is only one keypress per letter. A built-in dictionary maps candidate words to a key sequence. Ideally, "beep" is entered with T9 as

```
23370
beep_
```

This is an over simplification for at least two reasons. One is that multiple words may match the key sequence. If so, they are offered in order of their frequency in the language. The other is that the display is unstable during entry of a word, because of the ambiguity. The following is a more accurate rendering of entering "beep" using T9:

```
     2 a
    23 be
   233 bed
  2337 beer
 2337n beds
2337nn adds
2337nnn bees
2337nnnn beep
2337nnnn0 beep_
```

Here, we see both the instability in the display during entry and the need to press "n" four times to reach "beep". Although "one-key" in "one-key with disambiguation" is not strictly true, the overhead for English is small. One estimate is that 95% of words in English can be entered unambiguously using T9 [23].

1.2 Research on Ambiguous Keyboards

There is substantial research on ambiguous keyboards, such as the phone keypad. A recent survey [19] identifies three types of such keyboards: Qwerty-like, phone-like, and reduced-key. Qwerty-like ambiguous keyboards leverage users' familiarity with the Qwerty letter arrangement. One example is the *Blackberry 8100* ("SureType") by Research In Motion. As seen in Figure 2, the letters are arranged along five columns, rather than the usual ten with Qwerty keyboards. This allows for bigger keys, which in turn improves user access to the keys. With 26 letters on 14 keys, there is less ambiguity.

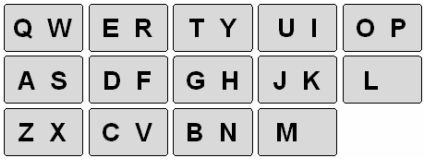


Figure 2. Letter-key assignments in the *Blackberry 8100*

Phone-like ambiguous keyboards combine the three-column format of phone keypads with a different letter arrangement. Generally, the goal is to increase the "linguistic separation" of keys by positioning common letters on different keys. This is seen in Ryu and Cruz's *LetterEase* in Figure 3a [21]. Letters are dispersed and assigned over the full 3×4 matrix of keys. An alternative is Gong and Tarasewich's *ACD* (alphabetically constrained design) in Figure 3b [6]. While the particular groupings were chosen to reduce ambiguity, the alphabetic arrangement aims to reduce users' visual scan time in finding the desired letter.

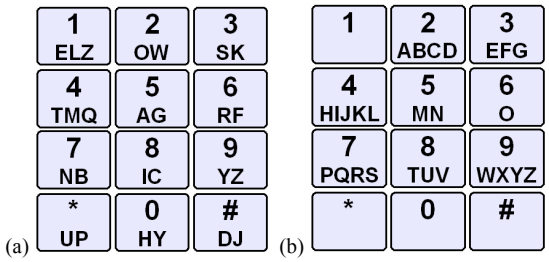


Figure 3. Phone-like ambiguous keyboards: (a) *LetterEase* [21], (b) *ACD* (alphabetically constrained design) [6]

Reduced-key ambiguous keyboards push the limits alluded to in the one-key challenge. With fewer keys, ambiguity increases. An example with four letters keys is Tanaka-Ishii et al.'s *TouchMeKey4* [25] in Figure 4a. The design uses an alphabetic letter ordering to reduce the visual scan time. Dunlop's four-key watch-top device in Figure 4b uses slightly different groupings [4]. Gong et al. describe a three-letter-key design [7] with letters also arranged alphabetically (Figure 4c).

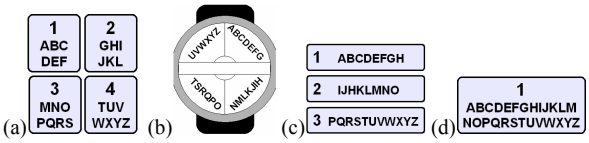


Figure 4. Small ambiguous keyboards: (a) *TouchMeKey4* [25], (b) watch-top [4], (c) L3K (letters on three keys) [7], (d) L1K (letters on one key)

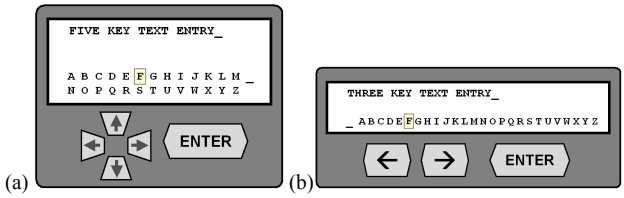
The L1K (letters on one key) design in Figure 4d is offered only to provoke discussion. Before considering L1K, we should mention that all the key arrangements discussed above require additional keys, such as a BACKSPACE key for error correction, a SPACE key to delimit words, or a "next" key to navigate through the ambiguous list, where necessary.

So, how would one enter text using the L1K keyboard in Figure 4d? The answer is simple: Using the same method as with the other keyboards: (i) press keys bearing the desired letters until the full word is entered, (ii) press "next" until the desired word in the candidate list is reached, then (iii) press SPACE to select the word. Of course, with the L1K design, entering a five-letter word means pressing the key five times, whereupon all five-letter words in the dictionary are candidates. If the user enters "which" – the most common five-letter word in English – the situation is fine (i.e., 111110). However, a word like "float" is likely to take about a thousand presses of "next", depending on the size of the dictionary (i.e., 11111nnn…0).

1.3 Direct vs. Indirect Input

The keyboards discussed above all use "direct input": keys are accessed directly by a finger, stylus, or mouse pointer. A small step forward in the one-key challenge is to separate input and

output. Letter keys become "virtual keys" on the system's display (output). Physical keys (input) indirectly move the focus point and select virtual keys. A two-way pager from the 1980s operated in this manner – the *AccessLink-2* by WirelessAccess (now defunct). The general idea is shown in Figure 5a where five physical keys navigate and select letters on virtual keys.



(a) (b)

Figure 5. Indirect text entry; (a) five keys (b) three keys

There are clear performance issues with this style of keyboard. How many focus movements are required, on average, for text entry in a given language? Can the number of focus movements be reduced using a different letter arrangement or some other technique? These questions are similar to those that arise in designing ambiguous keyboards.

Moving from five keys to three keys requires a linear sequence of letters. The arrangement in Figure 5b is sometimes called the "date stamp method". It has a history of use in video games, where players enter their name after achieving a high score.

To move from three keys to one key, another leap is required. Instead of using cursor keys to navigate virtual letter keys, navigating is automatic using a built-in timer. Letter keys are sequentially highlighted ("scanned") and when the desired letter key receives focus, it is selected using the sole physical key. The main problem with such "scanning keyboards" is that they are prohibitively slow for text entry. In the next section, we introduce some of the key design issues for scanning keyboards.

1.4 Scanning Keyboards

Users with physical disabilities are often unable to use a regular computer keyboard, a mobile phone keyboard, or a pointing device, such as a mouse. Input may be constrained to a single key or switch. Text entry is typically performed using a scanning keyboard. The general idea is shown in Figure 6a, for linear scanning. Focus advances from one virtual key to the next, paced by a programmable timer. When the desired letter key is highlighted, it is selected.

Whether the scanning interval is 5 seconds or 1 second, text entry using the scheme in Figure 6a is tedious. For example, it takes 26 scan steps to reach "z". For this reason, most scanning keyboards use some form of multi-level, or "divide and conquer" scanning, as shown in Figure 6b for row-column scanning. Here, scanning proceeds row to row. When the row bearing the desired letter is highlighted, it is selected. Scanning enters the row and proceeds from one letter key to the next. When the desired letter key is highlighted, it is selected. The selected letter is added to the text message and scanning reverts to the home position for input of the next letter.

Row-column scanning improves on linear scanning. The letter "j", for example, is reached in 5 scan steps (3 rows + 2 columns). Still, it is slow. Researchers have investigated many methods to speed-up text entry with scanning keyboards. The methods fall into three main categories: using different letter or row-column arrangements, using word or phrase prediction or completion, or reducing the scanning interval.

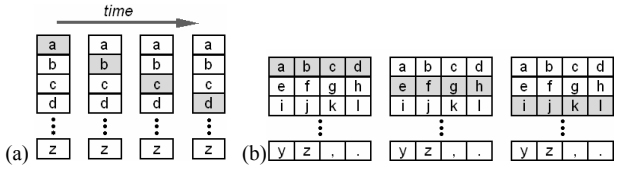


Figure 6. Scanning keyboards. (a) linear (b) row-column

The most obvious improvement for row-column scanning is to rearrange letters with frequent letters near the beginning of the scan sequence, such as in the first row or in the first position in a column. There are dozens of research papers investigating this idea, many dating to the 1970s or 1980s. Some of the more recent efforts are hereby cited [1, 9, 12, 15, 26]. Dynamic techniques have also been tried, whereby the position of letters varies depending on previous letters entered and the statistical properties of the language and previous text [12, 20].

Performance may also improve using a three-level selection scheme, also known as block, group, or quadrant scanning [3, 14]. The idea is to scan through a block of items (perhaps a group of rows). The first selection enters a block. Scanning then proceeds among smaller blocks within the selected block. The second selection enters one of the smaller blocks and the third selection chooses an item within that block. Block scanning is most effective if there are a large number of selectable items.

Reducing the scanning interval will increase the text entry rate, but there is a cost: higher error rates or missed opportunities for selection. One approach is to dynamically adjust the system's scanning interval. Decisions to increase or decrease the scanning interval can be based on previous user performance, including text entry throughput, error rate, or reaction time [11, 13, 24].

Word or phrase prediction or completion is also widely used with scanning keyboards [9, 20, 22]. As a word is entered, the current word stem is expanded to form a list of matching complete words. The list is displayed in a dedicated region with a mechanism for the user to select the word early.

We should be clear that all scanning keyboards in use are variations on "divide and conquer". For any given entry, the first selection chooses a group of items and the next selection chooses within the group. It is worth speculating on the prospect of combining the core concept of a scanning keyboard – input using one physical key – with the most appealing feature of an ambiguous keyboard – one keypress per character. We will do this shortly. First, we will summarize and examine published user evaluations of keyboards with just a few keys.

2. REVIEW OF USER STUDIES

The one-key challenge is not just about entering text using a single key, but doing so with reasonable speed. So, what is the state of the art on text entry speeds using just one, or a few, physical keys for input? As it turns out, most publications

Table 1
Comparison of Text Entry Studies Using a Small Number of Input Keys

Study (1st author)	Number of Keys[a]	Direct/ Indirect	Scanning	Number of Participants	Speed[b] (wpm)	Notes
Bellman [2]	5	Indirect	No	11	11	4 cursors keys + SELECT key. Error rates not reported. No error correction method.
Dunlop [4]	4	Direct	No	12	8.90	4 letter keys + SPACE key. Error rates reported as "very low".
Dunlop [5]	4	Direct	No	20	12	4 letter keys + 1 key for SPACE/NEXT. Error rates not reported. No error correction method.
Tanaka-Ishii [25]	3	Direct	No	8	12+	4 letters keys + 4 keys for editing, and selecting. 5 hours training. Error rates not reported. Errors corrected using CLEAR key.
Gong [7]	3	Direct	No	32	8.01	3 letter keys + two additional keys. Error rate = 2.1%. Errors corrected using DELETE key.
MacKenzie [16]	3	Indirect	No	10	9.61	2 cursor keys + SELECT key. Error rate = 2.2%. No error correction method.
Baljko [1]	2	Indirect	Yes	12	3.08	1 SELECT key + BACKSPACE key. 43 virtual keys. RC scanning. Same phrase entered 4 times. Error rate = 18.5%. Scanning interval = 750 ms.
Simpson [24]	1	Indirect	Yes	4	4.48	1 SELECT key. 26 virtual keys. RC scanning. Excluded trials with selection errors or missed selections. No error correction. Scanning interval = 525 ms at end of study.
Koester [10]	1	Indirect	Yes	3	7.2	1 SELECT key. 33 virtual keys. RC scanning with word prediction. Dictionary size not given. Virtual BACKSPACE key. 10 blocks of trials. Error rates not reported. Included trials with selection errors or missed selections. Fastest participant: 8.4 wpm.

[a] For "direct" entry, the value is the number of letter keys. For "indirect" entry, the value is the total number of keys.
[b] The entry speed cited is the highest of the values reported in each source, taken from the last block if multiple blocks.

focus on keystroke analyses or linguistic techniques. The research tends to be theoretical and, in most cases, the designs are not implemented and tested with users. Table 1 summarizes the results from nine papers where measurements with real users were performed and where text entry speeds were reported. Since we are interested in one-key text entry, the table is limited to methods using just one or a few keys. Evaluations using phone keypads and their variants are excluded.

The table includes four direct and five indirect entry methods. Three studies used scanning keyboards. Of the studies using 3-5 physical keys, text entry speeds were in the range of 8-12 wpm. Bellman and MacKenzie implemented a system similar to the five-key pager in Figure 5 [2]. They compared techniques to minimize focus movements by rearranging letters. A static arrangement was fastest, reporting in at about 11 wpm.

Dunlop's simulated watch-top text entry method placed letters on four keys in an alphabetic arrangement (Figure 4b) [4]. The implementation used a Hewlett Packard *iPAQ*. Entry rates were respectable at 8.9 wpm.[1] Dunlop and Masters evaluated a similar method using the same letter arrangement, but implemented on a Sony Ericsson *K300i* [5]. Entry speeds were 12 wpm, averaged over the last eight phrases of input. Both methods just cited are labeled "direct input" in the table, although this is not strictly true. Virtual letter keys were presented on the display with input using separate physical keys. However, since each physical key mapped directly to a virtual key, the direct-input category seems appropriate.

Tanaka-Ishii et al.'s *TouchMeKey4* system placed letters on three virtual keys (Figure 4a) [25]. Letter keys were selected using the mouse pointer. Separate + and – virtual keys moved through a list of candidate words and a SELECT key was clicked to choose the desired word. Entry speed was just over 12 wpm after five hours of practice. Gong et al. used linguistic modeling to improve the word list ordering in a similar system using three keys (Figure 4c) [7]. Entry speeds were about 8 wpm.

Three-key indirect text entry using ←, →, and SELECT keys and a linear array of letter keys was presented by MacKenzie (Figure 5b) [16]. Using a dynamic, optimized letter arrangement combined with a snap-to-home cursor mode yielded entry speeds just under 10 wpm.

Although there is a substantial body of research on scanning keyboards, only a few studies present a user evaluation where text entry speeds are reported. Three such studies appear along the bottom rows in Table 1. Baljlko and Tam describe a row-column scanning system where letters of the alphabet and other symbols were arranged according to a Huffman encoding tree [1]. One key made selections while a BACKSPACE key corrected errors. The fastest entry speeds were about 3 wpm.

Simpson and Koester used a conventional row-column scanning arrangement, similar to Figure 6b, while automatically varying the scanning interval according to user performance [24]. The system included one physical key for selecting, but no mechanism to correct errors. The highest entry rate was 22.41

[1] Text entry speed was reported as "time per phrase". The values were converted to "words to minute" using information provided by the author [Dunlop, personal communication].

characters per minute (÷5 = 4.48 wpm),[2] averaged over four participants on the 4th of four blocks of trials. The scanning interval in the last block was 525 ms. Importantly, the text entry speed calculation excluded both selection errors and letter selections that did not occur at the earliest opportunity during scanning. No doubt, the entry rate would be lower – perhaps, quite a bit lower – if all the data were included.

A higher entry speed was reported by Koester and Levine [10] who compared a letters-only and letters+WP (word prediction) row-column scanning system with single switch input. After 20 blocks of trials, the entry speed was fastest using letters+WP. The average entry speed was 7.2 wpm, with one participant reaching 8.4 wpm. Only three participants were used and error rates were not reported (although errors were ≈5% during the practice trials). The dictionary size was also not reported. A small dictionary improves the performance since words are more likely to appear in prominent positions in the candidate list. Nevertheless, Koester and Levine's one-key text entry rate of 7.2 wpm (average over three participants) or 8.4 wpm (fastest participant) is the fastest in the literature for one-key text entry. Thus, the bar is set for the one-key challenge.

3. SCANNING AMBIGUOUS KEYBOARD

All the pieces for a new one-key text entry method exist in the paragraphs above. It is just a matter of putting them together. A "scanning ambiguous keyboard" (SAK) combines the most demanding requirement of a scanning keyboard – input using one key – with the most appealing feature of an ambiguous keyboard – one keypress per character. The general idea is shown in Figure 7. Letters are assigned over a small number of virtual keys in a letter-selection region. Scanning proceeds left to right in a cyclic pattern. When the key bearing the desired letter is highlighted, it is selected. There is only one selection per letter and only one physical key (not shown). With each letter selection, a list of candidate words is presented.

When the full word is entered (or earlier, see below), SPACE is selected to begin scanning in the word-selection region. When the desired word is highlighted, it is selected and added to the text message with a terminating space. Scanning then returns to the letter-selection region for entry of the next word.

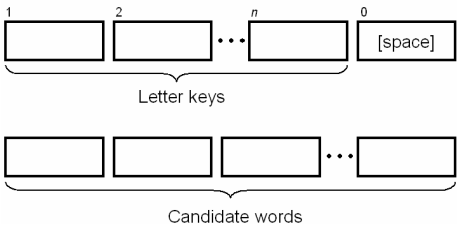


Figure 7. A scanning ambiguous keyboard (SAK) combines letter-selection (top) and word-selection (bottom) regions

The candidate list is organized in two parts. The first presents words exactly matching the current keycode sequence, ordered by their frequency in the language. The second presents extended words, where the current keycode sequence is treated as the word stem.

Of course, there are many design issues. The first is simple: How many letter keys should be used and how should letters be assigned to keys? In answering this question, a model was built to search for designs that minimize *SPC* – the number of scan steps per character required, on average, for text entry in a given language.[3] *SPC* includes both passive scan steps (no selection) and active scan steps (selection) and also includes the number of scan steps to reach and select the desired word in the candidate list.

The model includes the possibility of "early selection". If the candidate word appears in the list before the word is fully entered, it is selected – if the result is fewer scan steps compared to the alternative of continuing to select letter keys.

A novel feature of SAK designs is *double selection*. Two selections can occur in the same scanning interval if the highlighted key bears both the desired letter and the next letter. Of course, the ability to double-select depends on the scanning interval and a user's motor ability.

Clearly, SAK designs are compromises. Fewer letter keys reduce the scans in the letter selection region but increase ambiguity and, therefore, increase the scans in the word selection region. More letter keys and the opposite occurs.

The model works with a language corpus in the form of a word-frequency list. For this, we used Silfverberg et al.'s list of 9,025 unique words and frequencies, drawn from the British National Corpus [23].

An exhaustive run of the model computed *SPC* for all designs from 1 to 6 virtual letter keys with all possible alphabetic letter assignments. The letter assignment with the lowest scan steps per character, *SPC* = 1.834, is shown in Figure 8.

SPC < 2 is remarkably low. As an example, the word "character" can be entered in just 15 scan steps:

```
c...a.r.a.tS..W
```

A lowercase letter is a selection on the key bearing the letter. A period (".") is a passive scan step. If double selection is possible, only the first letter is shown. In the example, "c" = "ch" and the second "a" = "ac". "S" is a selection on the virtual SPACE key. "W" is a word selection. Early selection is shown in the example, since all letters were not entered. Evidently "character" appeared as the 3rd word in the candidate list after "charact" (1113113) was entered. For the example word, *SPC* = 1.50, computed as 15 scan steps divided by 10 characters ("character_").

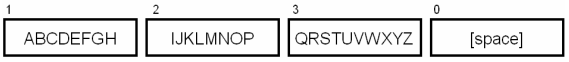


Figure 8. Scanning ambiguous keyboard (SAK) letter assignments with the lowest scan steps per character (*SPC*).

The *SPC* calculation, whether for an example word or English in general, is a best case, and assumes users take all opportunities to optimise. Of course, missed opportunities can and will occur,

[2] It has been a convention dating back to the 1920s to define "word" in "words per minute" as five characters, including letters, punctuation, etc. [27].

[3] The full details of the model are described in a separate paper [17].

depending on a variety of factors, such as the user's skill, motor ability, or the scanning interval. But, missed opportunities are not errors; they simply slow the entry of words and phrases.

Two designs in the literature are similar to the SAK concept presented here; however, both involve more than one physical key. Harbusch and Kühn describe an ambiguous keyboard using "step scanning": one key advances the focus point while another makes selections [8]. Venkatagiri describes a scanning keyboard with three or four letters per virtual key [26]. Separate physical keys choose the desired letter once a key receives focus. Both papers just cited present models only. No user studies were performed, nor were the designs actually implemented.

3.1 SAK Implementation

A prototype SAK application was developed in Java to implement the scanning ambiguous keyboard described above. A screen snap is shown in Figure 9. The scanning interval is set by a configuration parameter. Physical input uses any key on the system's keyboard. The prototype presents text phrases to the user (top field). As the user enters the phrase, the current key codes and transcribed text appear in separate fields. Both keystroke and phrase-level data are collected and stored.

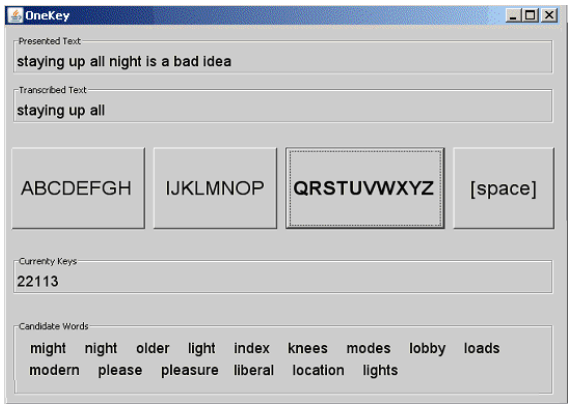


Figure 9. Scanning Ambiguous Keyboard (SAK) implementation

In the screen snap, the user has entered the first three words in the phrase "staying up all night is a bad idea". The next word, "night" has been completely entered (keycodes = 22113). Focus is on the third letter key. The desired word is the second entry in the candidate list. A selection in the next scan interval ("[space]") transfers scanning to the word-selection region. A pause then select chooses the word and adds it to the transcribed text.

3.1.1 Error Correction

It is important to support error correction with any text entry method. One possibility here is to add a separate physical key to delete the last selection [1]. Of course, this compromises the one-key constraint. Another possibility is to add a virtual DELETE or BACKSPACE key among the scanned letter keys. Most scanning keyboards do this [12, 14, 15, 26]. This is not feasible for SAK designs, however, since a small set of letter keys is important to keep *SPC* low and thereby increase text entry speed.

Error correction was implemented in the prototype SAK using a "long press" – pressing and holding the physical key for two or more scan steps. There are at least two examples of this for scanning keyboards [9, 20]. If a long press occurs during entry of a word, the current key sequence is cleared. If a long press occurs between words, the last word is deleted.

3.1.2 Ambiguity Analysis

The prospect of very long candidate lists is unavoidable with ambiguous keyboards having just a few keys [19]. Although the *SPC* calculation accounts for this, it does so only as an overall average for text entry in a given language. And so, we ask: How long are the candidate lists? Are long candidate lists common? The answers to these questions are determined by two factors: the letter-key assignments and the dictionary. For the 9,025-word dictionary used with the prototype SAK, 55.7% of the words are at the front of the candidate list when word selection begins. Furthermore, 82.2% of the words appear in the candidate list at position 4 or better; 94.8% are at position 10 or better, and 99.6% are at position 20 or better.

So, very long candidate lists are rare. But, still, they occasionally occur. As an example of position 20, if a user wished to enter "alas" (1213) it would be at the end of a long list of candidates: { does, body, goes, boat, dogs, diet, flat, coat, andy, gift, flew, ends, aids, alex, clay, bias, blew, gods, dies, alas }.

While models are excellent tools for exploring human-computer interfaces, it remains to see how the prototype SAK will fare in a user evaluation.

4. USER EVALUATION

Twelve participants were recruited from the local university campus to enter text on the scanning ambiguous keyboard described above. The mean age was 25.3 years (*SD* = 3.2). Five were female, seven male.

After a briefing and demonstration, participants were allowed to enter a few practice phrases. Following this, participants entered five blocks of phrases, five phrases per block. The total time was about one hour per participant.

The application pulled phrases at random from a standard set of 500 phrases [18]. The computed *SPC* for the phrase set was 1.980. Given that *SPC* = 1.834 for the SAK under test, the phrases were, on average, slightly more complicated than English in general. This is an important point. It would be relatively easy to concoct text phrases using words with low *SPC*s and thereby obtain inflated text entry speeds. This was not done. The scanning interval was set to 1100 ms for the 1st block. It was decreased by 100 ms per block, finishing at 700 ms.

4.1 Results and Discussion

The results are shown in Figure 10. With all trials included, entry speed improved from 3.98 wpm in the 1st block to 5.11 wpm in the 5th block.

Word corrections (long presses) occurred at a rate of about two per phrase, which is rather high. However, the result was transcribed text with a very low error rate of only 0.96% (i.e., accuracy > 99%).

Although the main effect of block on entry speed was significant ($F_{4,44} = 7.58$, $p < .0001$), this is expected in view of the confounding influence of scanning interval. In fact, the evaluation was designed specifically to elicit an improvement in text entry speed by starting with a conservative scanning interval and speeding up gradually from block to block.

The results above are promising for one-key text entry. In fact, even faster rates may be possible with some design modifications (see below).

Note that the "all trials" line in Figure 10 shows no improvement in text entry speed after the 3rd block. The main reason for this was the faster scanning intervals, which in turn resulted in more selection errors or missed opportunities. Each block involved only 8-10 minutes of text entry, so it is likely the scanning interval was reduced a bit too soon for participants to adjust in the 4th and 5th blocks.

Given that the paper by Simpson and Koester in Table 1 reported text entry speed for error-free trials only [24], an additional "error-free trials" line is added to Figure 10. For this data set, entry speed increased from 5.30 wpm in the 1st block to 7.03 wpm in the 5th block.

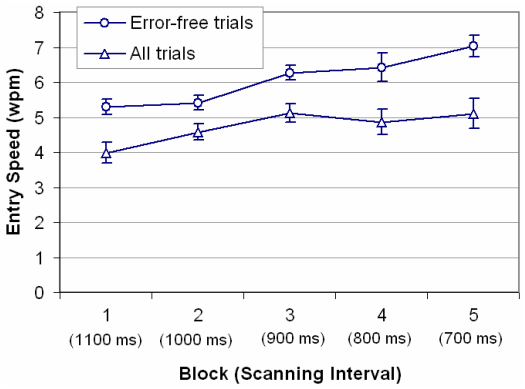


Figure 10. Entry speed (wpm) by block and scanning interval. Results are shown for all trials (bottom) and error-free trials (top). Error bars show ±1 standard error.

4.1.1 Timer Restart On Selection

Participants made a variety of comments and suggestions about the interface. One idea to improve interaction was to restart the scanning timer when a selection is made. This would extend the current scanning interval and provide the user with precious time to think and to prepare for the next selection. Taken alone, restarting the timer slows the interaction. But, the net effect may be to increase text entry speed if there are fewer selection errors or missed opportunities. Restarting the timer also means that faster scanning intervals become possible, because, in any interval where there is a selection, the duration of the scan step is automatically extended.

An interesting side effect of restarting the timer is that more than two selections are possible in a single scanning interval. In fact it would be simple to make, say, four or five selections since with each selection the entire scanning interval is again available for the next key press. Thus, the *double selection* input strategy noted earlier becomes *multiple selection*.

A simple script was run to determine the prevalence of such multiple-select opportunities in the 9025-word dictionary. As it turns out, a full 36% of the words offer at least one opportunity for a triple selection (e.g., "book" = 1222). There are also many words with >3 consecutive letters on the same key, and even a few with numerous consecutive letters on the same key (e.g., "feedback" = 11111112).

This idea was considered so provocative, the SAK application was modified to implement a "timer restart on selection" mode. One participant agreed to do an extra five blocks (five phrases each) to test the mode. For these "extended" trials, the scanning interval started at 700 ms and was reduced by 50 ms per block, finishing at 500 ms. The results are shown in Figure 11.

The entry speeds ranged from 7.38 wpm in the 6th block to 9.28 wpm in the 10th block. The overall error rate for the extended trails was very low, at 0.81%. A look at the raw data revealed some use of multiple selections. Of 558 total letter selections, there were 98 (17.5%) double selections, 17 (3.0%) triple selections, and 3 (0.5%) quadruple selections.

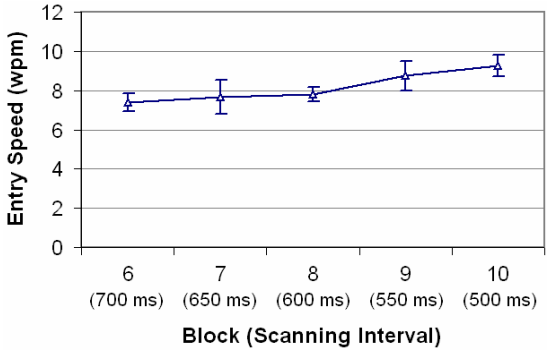


Figure 11. Extended trials for one participant using the "timer restart on selection" mode. The data are for five blocks with five phrases per block.

Comparing the results to those in Table 1 for one-key input is promising. The SAK prototype faired considerably better than Baljko and Tam's row-column method using a Huffman encoding tree (3.08 wpm) [1] and also better than Simpson and Koester's row-column design with an adaptive scanning interval (4.48 wpm) [24]. At 7.2 wpm, Koester and Levine's letters+WP system compares quite well [10]. However, they tested three participants only, practiced over 20 blocks. The results here are for 12 participants over just 5 blocks. Koester and Levine cite their best participant's performance as 8.2 wpm. In Figure 11, a participant reached 9.28 wpm on the 10th block using the "timer restart upon selection" mode for blocks 6-10. Thus, the best participant's performance with SAK is about 13% faster than Koester and Simpson's best participant. No doubt a more extended test of the SAK design over, say, 20 blocks with the "timer restart on selection" mode would yield even better results.

4.1.2 Non-dictionary Words and Other Features

The SAK design tested here is a research prototype. Further development is needed, for example, to include a spell mode to add new words to the system's dictionary. Other features are also needed, such as adding punctuation, changing configuration parameters, or switching applications. One possibility is to use the long press as a conduit. A long press could still be used for error correction, but the design could be re-worked to popup a small set of scanned virtual "option keys" (see Figure 12).

Delete Last Letter	Delete Last Word	Spell Mode	Punctuation Mode	Next Application	Setup	Return

Figure 12. Options accessed through a long press

5. CONCLUSIONS

This paper laid out the one-key challenge for text entry. A new text entry method using just one physical key for input was presented, implemented, and tested with users. The design is a scanning ambiguous keyboard with an alphabetic arrangement of letters on three keys. The design is useful for accessible computing in situations where one-key input is needed.

After just a small amount of practice (<1 hour), the average text entry rate for 12 users was 5.11 wpm, including all data, or 7.03 wpm considering error-free trials only. Using a "timer restart on selection" mode, one participant performed an additional five blocks of trials (five phrases per block) and achieved a rate of 9.28 wpm on the last block.

6. REFERENCES

1. Baljko, M. and Tam, A., Indirect text entry using one or two keys, *Proc ASSETS 2006*, (ACM, 2006), 18-25.
2. Bellman, T. and MacKenzie, I. S., A probabilistic character layout strategy for mobile text entry, *Proc Graphics Interface '98*, (Toronto: Canadian Information Processing Society, 1998), 168-176.
3. Bhattacharya, S., Samanta, D., and Basu, A., User errors on scanning keyboards: Empirical study, model and design principles, *Interacting with Computers*, *20*, 2008, 406-418.
4. Dunlop, M. D., Watch-top text-entry: Can phone-style predictive text entry work with only 5 buttons?, *Proc MobileHCI 2004*, (Heidelberg, Germany: Springer-Verlag, 2004), 342-346.
5. Dunlop, M. D. and Masters, M. M., Investigating five key predictive text entry with combined distance and keystroke modelling, *Pervasive and Ubiquitous Computing*, *12*, 2008, 589-598.
6. Gong, J. and Tarasewich, P., Alphabetically constrained keypad designs for text entry on mobile phones, *Proc CHI 2005*, (New York: ACM, 2005), 211-220.
7. Gong, J., Tarasewich, P., and MacKenzie, I. S., Improved word list ordering for text entry on ambiguous keyboards, *Proc NordiCHI 2008*, (New York: ACM, 2008), 152-161.
8. Harbusch, K. and Kühn, M., An evaluation study of two-button scanning with ambiguous keyboards, *Proc AAATE 2003*, (Taastrup, Denmark: AAATE c/o Danish Centre for Assistive Technology, 2003), 954-958.
9. Jones, P. E., Virtual keyboard with scanning and augmented by prediction, *Proc 2nd European Conference on Disability, Virtual Reality and Associated Technologies*, (University of Reading, UK, 1998), 45-51.
10. Koester, H. H. and Levine, S., Learning and performance of able-bodied individuals using scanning systems with and without word prediction, *Assistive Technology*, *6*, 1994, 42-53.
11. Lesher, G., Higginbotham, D. J., and Moulton, B. J., Techniques for automatically updating scanning delays, *Proc RESNA 2000*, (Arlingtion, VA: RESNA, 2000), 85-87.
12. Lesher, G., Moulton, B., and Higginbotham, D. J., Techniques for augmenting scanning communication, *Augmentative and Alternative Communication*, *14*, 1998, 81-101.
13. Lesher, G., Moulton, B., Higginbotham, J., and Brenna, A., Acquisition of scanning skills: The use of an adaptive scanning delay algorithm across four scanning displays, *Proc RESNA 2002*, (Arlington, VA: RESNA, 2002), 75-77.
14. Lin, Y.-L., Chen, M.-C., Wu, Y.-P., Yeh, Y.-M., and Wang, H.-P., A flexible on-screen keyboard: Dynamically adapting for individuals needs, *Universal Access in Human-Computer Interaction. Applications and Services*, (Berlin: Springer, 2007), 371-379.
15. Lin, Y.-L., Wu, T.-F., Chen, M.-C., Yeh, Y.-M., and Wang, H.-P., Designing a scanning on-screen keyboard for people with severe motor disabilities, *Computers Helping People With Special Needs*, (Berlin: Springer, 2008), 1184-1187.
16. MacKenzie, I. S., Mobile text entry using three keys, *Proc NordiCHI 2002*, (New York: ACM, 2002), 27-34.
17. MacKenzie, I. S., SAK: Scanning ambiguous keyboard for efficient one-key text entry, *(under review)*, 2008.
18. MacKenzie, I. S. and Soukoreff, R. W., Phrase sets for evaluating text entry techniques, *Ext Abstracts CHI 2003*, (New York: ACM, 2003), 754-755.
19. MacKenzie, I. S. and Tanaka-Ishii, K., Text entry with a small number of buttons, in *Text entry systems: Mobility, accessibility, universality*, (I. S. MacKenzie & Tanaka-Ishii, K., Eds.). San Francisco, Morgan Kaufmann, 2007, 105-121.
20. Miró, J. and Bernabeu, P. A., Text entry system based on a minimal scan matrix for severely physically handicapped people, *Proc ICCHP 2008*, (Berlin: Springer, 2008), 1216-1219.
21. Ryu, H. and Cruz, K., LetterEase: Improving text entry on a handheld device via letter reassignment, *Proc OZCHI 2005*, (New York: ACM, 2005), 1-10.
22. Shein, F., Hamann, G., Brownlow, N., Treviranus, J., Milner, D., and Parnes, P., WiVik: A visual keyboard for Windows 3.0, *Proc RESNA 1991*, (Arlington, VA: RESNA, 1991), 160-162.
23. Silfverberg, M., MacKenzie, I. S., and Korhonen, P., Predicting text entry speed on mobile phones, *Proc CHI 2000*, (New York: ACM, 2000), 9-16.
24. Simpson, R. C. and Koester, H. H., Adaptive one-switch row-column scanning, *IEEE Trans Rehabilitation Engineering*, *7*, 1999, 464-473.
25. Tanaka-Ishii, K., Inutsuka, Y., and Takeichi, M., Entering text with a four-button device, *Proc Conference on Computational Linguistics - Vol. 1*, (Morristown, NJ: Association for Computational Linguistics, 2002), 1-7.
26. Venkatagiri, H. S., Efficient keyboard layouts for sequential access in augmentative and alternative communication, *Augmentative and Alternative Communication*, *15*, 1999, 126-134.
27. Yamada, H., A historical study of typewriters and typing methods: From the position of planning Japanese parallels, *Journal of Information Processing*, *2*, 1980, 175-202.