5 CHAPTER

Text Entry Using a Small Number of Buttons

I. Scott MacKenzie York University, Toronto, ON, Canada
Kumiko Tanaka-Ishii University of Tokyo, Tokyo, Japan

5.1 INTRODUCTION

This chapter focuses on systems in which the user enters text by pressing keys or buttons and in which more than one character or letter is assigned to each button. Such keyboards are ambiguous because there is uncertainty as to the intended symbol when a key is pressed. There is recent worldwide interest in such *ambiguous keyboards* because of mobile computing, for which space is limited. Also, such keyboards widen the communications possibility for users with physical disabilities who have insufficient motor facility to operate a full-size keyboard (see Chap. 15).

Even though ambiguous keyboards have a reduced complement of keys, users still require access to a large set of characters, including the alphabet, numbers, symbols, and editing keys. Bear in mind that the standard PC keyboard has about 100 keys, yet can produce closer to 800 symbols with the use of modifier keys such as SHIFT, CONTROL, or ALT. So, in this sense, the standard PC keyboard is also ambiguous.

Two general methods enable access to a large set of characters; these differ depending on who performs the disambiguation. First, there is the *multi-tap method* or *non-predictive method,* in which the user disambiguates using multiple strokes to uniquely indicate a character. In the case of a full-size keyboard, additional strokes such as those applied through the CONTROL or SHIFT key are a kind of multi-tap entry. Although the multi-tap method is as old as the typewriter, the concept received new attention when text entry on a phone keypad was first considered in the 1970s (Smith & Goodwin, 1971).

The second approach uses a *predictive method,* in which the system disambiguates and presents a list of ordered candidates from which the user chooses. Prediction in English was first studied in the 1950s as a general problem in information theory (Shannon, 1951) and somewhat later in 1970s as applied to text entry using a phone keypad (Desautels & Soffer, 1974; Rabiner & Schafer, 1976). In addition, predictive entry methods for English were proposed in the 1980s as a typing aid (Darragh *et al.*, 1992). The first concept of text entry by prediction, though, occurred in the 1960s in

Japan by Kurihara (Kurihara & Kurosaki, 1967; Kurihara, 1970), where the language uses thousands of characters. When computers were introduced in Japan, the first problem was how to enter thousands of characters using a keyboard designed for European languages. Entry by prediction is now widely used throughout Japan and China and is spreading to many applications in European languages.

Conceptually, key ambiguity lies in a continuum (see Fig. 5.1). At one extreme, we have a keyboard with a dedicated key for each symbol in the language (Fig. 5.1a),

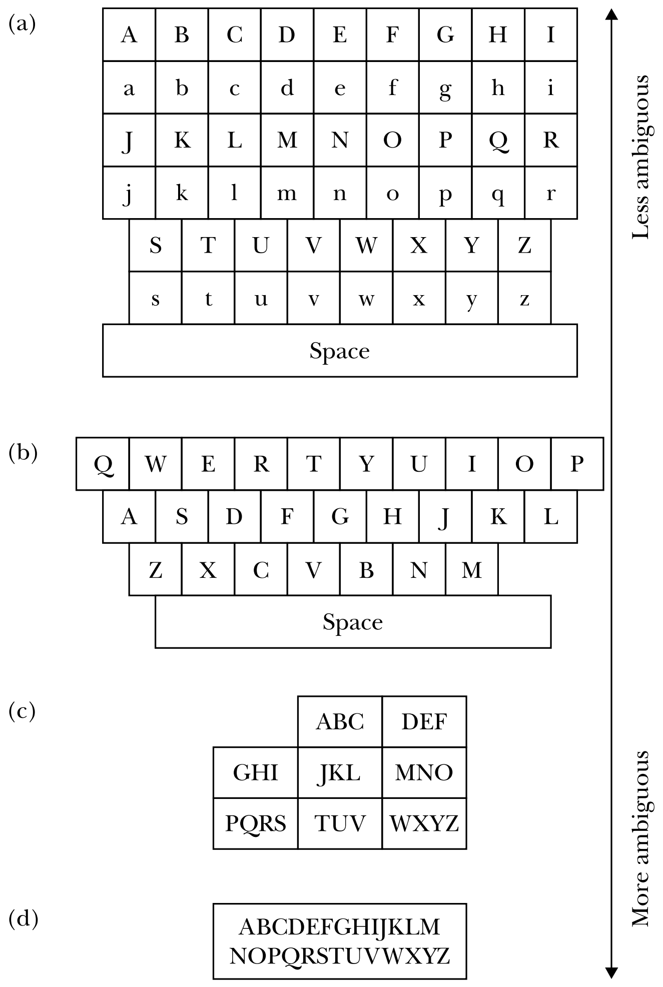

FIGURE 5.1 Key-ambiguity continuum. (a) Fictitious alphabetic keyboard with distinct keys for upper- and lower-case letters. (b) Qwerty keyboard. (c) Standard telephone keypad. (d) Hypothetical single-key keyboard, which, to be useful, would require either many mode keys or a near-psychic disambiguation algorithm.

while at the other we have just one key that maps to every symbol in the language (Fig. 5.1d). The Qwerty keyboard (Fig. 5.1b) and telephone keypad (Fig. 5.1c) represent two relevant points in the continuum.

This continuum suggests many other possibilities in keyboard design. Basically, two decisions are required: how many keys to include and the assignment of symbols to keys. Usually, the number of keys (or buttons in devices like cell phones) depends on the physical limits of the device. The second decision is more critical. Clearly, the assignment governs the entry efficiency regardless of whether the entry method is predictive or non-predictive.

This chapter introduces various ambiguous keyboards and explores how the efficiency of such keyboards is measured. To this end, we describe text entry methods for mobile phones and then present a general measure of each method's efficiency.

5.2 MOBILE PHONE KEYPAD AND ENTRY METHODS

With about 1 billion SMS messages sent per day, today's mobile phone keypad (see Fig. 5.2) is one of the world's most common devices for text entry.

The 12-key keypad consists of number keys 0–9 and two additional keys (* and #). Characters A–Z are spread over keys 2–9 in alphabetic order. The placement of

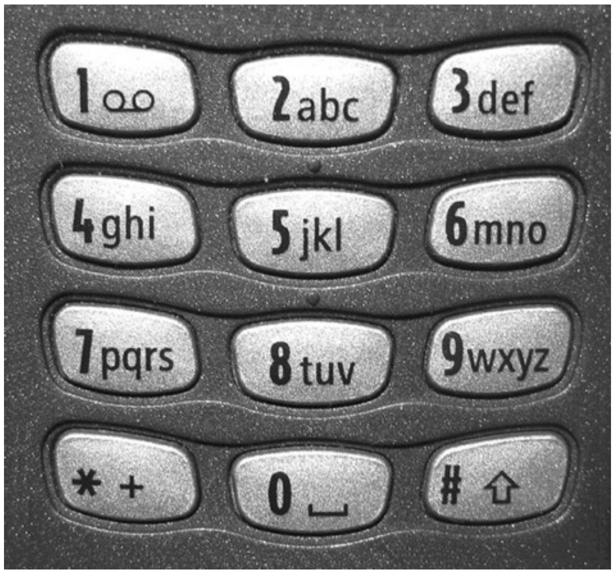

FIGURE 5.2 Standard 12-key telephone keypad. The 26 letters of the English alphabet are assigned to keys 2 through 9. The SPACE character is typically assigned to the 0 key.

characters is similar on most mobile phones, as it is based on an international standard. The SPACE character is typically assigned to the 0 key or sometimes to the # key. Since there are fewer keys than the 26 needed for the letters A–Z, three or four letters are grouped on each key and, so, ambiguity arises. As noted above, text entry using a mobile phone keypad is possible using non-predictive or predictive methods. With a multi-tap, or non-predictive method, the user disambiguates the meaning of the entry through multiple keystrokes.

With a typical multi-tap method in English, the user presses each key one or more times to specify the input character. For example, the 2 key is pressed once for A, twice for B, and three times for C. There is the additional problem of segmentation, when a character is on the same key as the previous character, for example, the word "ON" because both O and N are on the 6 key. To enter "ON" the user presses 6 three times for "O", segments, then presses 6 twice more for "N". Segmentation is performed using a timeout (e.g., 1.5 seconds on Nokia phones) or by pressing a "timeout kill" key (e.g., the down-arrow key). The user decides which strategy to use.

A multi-tap method can be designed in another way. With the *matrix* method, each character requires two strokes, the first to select the desired character, and the second to indicate the rank of that character on the key. For example, to enter C, the user presses 2 first and then 3, indicating that C is the third character on key 2. Thus, for every character, the user presses exactly two keys. This method requires that the number of characters assigned to a key is less than the total number of keys. The matrix method is a common method for Japanese entry on mobile phones.

The second way to enter characters is by prediction. In this method, disambiguation is performed first by the system and the user then selects the target from candidates shown by the computer. This is realized by providing the system with linguistic knowledge via a large amount of text data. Specifically, the user enters text by repeating the following process:

Step 1. The user enters a *code* sequence of the target text.

Step 2. The text entry system *decodes* the code by looking for corresponding targets in a dictionary associated with the software. It sorts the candidates in a relevant order using linguistic knowledge and displays them for the user. (The technological aspects of ordering candidates are explained in detail in Chap. 2.)

Step 3. The user chooses the target from among the candidates.

A well-known method within this framework is *one-key with disambiguation*. The technique has been used since the 1970s (e.g., Rabiner & Schafer, 1976), although today it is most commonly associated with *T9* by Tegic Communications, Inc. (www.tegic.com). With *T9*, each key is pressed only once. For example, to enter "THE," the user enters 8-4-3-0 (see Fig. 5.2). The 0 key, for SPACE, delimits words and terminates disambiguation of the preceding keys. *T9* compares the word possibilities to a linguistic database to guess the intended word.

Naturally, multiple words may have the same key sequence. In these cases the most common is the default. A simple example follows using the well-known “quick brown fox” phrase (words are shown top to bottom, most probable at the top):

843	78425	27696	369	58677	6837	843	5299	364
the	**quick**	**brown**	**fox**	**jumps**	**over**	**the**	jazz	**dog**
tie	stick	crown		lumps	muds	tie	**lazy**	fog
vie						vie		

Of the nine words in the phrase, eight are ambiguous, given the key sequence. For seven of the eight, however, the intended word is the most probable word. The intended word is not the most probable word just once, with “jazz” being more probable in English than “lazy.” In this case, the user must press a “NEXT” key (e.g., down arrow) to obtain the desired word (i.e., lazy = 5299N0). Evidently, the term “one-key” in “one-key with disambiguation” is an oversimplification!

Another commonly used method using prediction is *entry by completion.* Entry by completion is associated with commercial software, such as available from CIC Corp. (www.cic.com) or ZI Corp. (www.zicorp.com). Completion is also common in mobile phone entry systems in East Asian countries.

As an example of entry by completion, when entering “quick,” instead of entering the whole word, the user may enter only “qui” and then the whole word is guessed by the system. The candidate list in this case would include words other than “quick,” such as “quit” and “quiche.”

Entry by completion can be combined with one-key with disambiguation. In the case of “quick,” the user proceeds with the intention of entering 7-8-4-2-5, but receives complete-word candidates as entry proceeds, for example, after entering 7-8-4 or even 7 only. Entering 7-8-4 results in multiple words that begin with this combination: “still,” “suggest,” “quite,” “sugar,” “suit,” “stick,” etc. Note that ambiguity is greater when completion is used. Still, if the guesses are good, the user might need fewer keystrokes: the number of keystrokes may be less than the length of the word.

Generally, in both non-predictive and predictive methods, there is a trade-off between the degree of ambiguity and entry efficiency, with efficiency governed by the keyboard design. For example, if a phone keypad assigned A through Q to the digit 0 and R through Z to keys 1 to 9, respectively, the entry efficiency will be further limited no matter which entry method is adopted. To help clarify and quantify the trade-off, we next introduce evaluation metrics.

5.3 CHARACTERISTIC MEASURES FOR AMBIGUOUS KEYBOARDS

Design is defined as achieving goals within constraints (Dix *et al.*, 2004, p. 193). The “goal” in designing text entry methods is always to create a “better” method. Since empirical evaluation with users always follows design (sometimes occurring substantially later or not at all; see examples at the beginning of Chap. 4), researchers often use a model for the conjectured benefits of a design. For the research leading to the designs described in this section, the model hinges on a statistic—such as KSPC (MacKenzie, 2002), keystroke savings (Higginbotham, 1992; Koester & Levine, 1998), or word collisions (www.eatoni.com)—that captures the benefits in terms of the keystroke requirements of the technique. We here introduce several measures based on statistics.

The first is KSPC/KSPW, an acronym for "keystrokes per character/word." KSPC and KSPW are metrics to characterize keyboards with reduced keys or those using predictive aids[1]. KSPC and KSPW represents the number of key presses, or keystrokes, on average, to produce a character or word of text on a given keyboard using a given interaction technique in a given language.

Specifically,

$$\text{KSPC} = \Sigma_c \, K(c) P(c), \tag{5.1}$$

where $P(c)$ describes the probability of characters and $K(c)$ describes the number of keystrokes required to enter a character.

When a predictive method is used, both KSPC and KSPW can be calculated. In Japanese and Chinese, there are phrase-based entry systems (see Chap. 11), but here we limit the discussion to the most commonly used word-based entry. In the case of KSPW, the definition is

$$\text{KSPW} = \Sigma_w \, K(w) P(w), \tag{5.2}$$

where $K(w)$ indicates the number of keystrokes required to enter w and $P(w)$ indicates the probability of word w (Tanaka-Ishii *et al.,* 2002). This KSPW is transformed into $\text{KSPC}_{\text{word}}$ measured by word units (MacKenzie, 2002) by dividing KSPW by $\Sigma_w |w| \, P(w)$, where $|w|$ denotes the length of w. In both cases, $K(w)$ is further divided into two stages of entry: the first corresponds to the word typeface (for example, 5-2-9-9 to enter "lazy") and the other to the selection of the word (two scans for the word "lazy" in our "quick brown fox" example above). We use $K_{\text{code}}(w)$ to denote the former and $K_{\text{scan}}(w)$ to denote the latter.

One benefit of $\text{KSPC}_{\text{word}}$ for the word case is that it can be compared with non-predictive methods. On the other hand, KSPW allows us to determine the per-word entry efficiency, which is more direct in predictive entry by word units.

Both $\text{KSPC}_{\text{word}}$ and KSPW are somewhat simplistic because neither reflects the attention demands or movement requirements of the technique. Such a problem is

[1] The term "KSPC" is used here with a meaning slightly different from that in Chap. 3. As used here, KSPC (as well as KSPW, K_{code}, and K_{scan}) is a *characteristic measure,* relating to inherent properties of a keyboard. KSPC, as used in Chap. 3, is a *performance measure,* relating to the keystrokes actually used in producing text.

partly solved by separately analyzing $K_{\text{code}}(w)$ and $K_{\text{scan}}(w)$, thus clarifying the degree of ambiguity upon entry. Per-word $K_{\text{scan}}(w)$ on average represents the average ranking of the target word:

$$K_{\text{scan}} = \Sigma_w K_{\text{scan}}(w) P(w). \tag{5.3}$$

This forms part of Formula (5.2) (the rest is the per-word average of $K_{\text{code}}(w)$, denoted as K_{code} in the following). This average ranking can be measured more finely by the average complexity of word W (for example, "lazy"), predicted given an entry code T (for example, 5299). Such complexity is formally defined as conditional entropy (Bell *et al.,* 1990; Tanaka-Ishii *et al.*, 2002):

$$H(W|T) = -\Sigma_{w,t} P(w,t) \log P(w|t), \tag{5.4}$$

where t denotes the entry to get the word w. $H(W|T)$ represents the number of bits required to encode W given a user entry T. Therefore, the smaller $H(W|T)$ is, the less the effort to choose W. When there is no ambiguity, $H(W|T)$ is 0.0.

Calculating the measures above requires a keystroke model for the interaction method and also a language model to estimate the word and character probabilities. This is typically in the form of a token-frequency list appended with the required keystrokes to enter each word. The tokens may be words, single letters, digrams, trigrams, and so on; however, a word-frequency list is the most convenient, particularly for predictive entry techniques (see Chap. 2). The list is a reduction of a corpus and is, arguably, "representative" of a language. For the calculations presented in this chapter, we used a word-frequency list derived from the British National Corpus, the same list described by Silfverberg *et al.* (2000). The list contains the 9022 most common words in English, with frequencies totaling 67,962,112.

5.4 MOBILE PHONE KEYPAD VARIANTS

Due to the SMS messaging phenomenon noted earlier and to the difficulties in entering text on a mobile phone keypad, substantial research has emerged in recent years on alternative text entry techniques with reduced-key keyboards. Some examples are given in Fig. 5.3.

The phone keypad in Fig. 5.3a is "Qwerty-like." Hwang and Lee (2005) assigned letters to keys while trying to mimic the arrangement in Qwerty keyboards. The rationale is that users require less visual scan time to find letters if the arrangement is familiar. This, combined with placing letters on nine keys (rather than eight, as on a phone keypad) and slightly reworking the assignment, is intended to improve the disambiguation process and, hence, improve performance. Ryu and Cruz's (2005) efforts in Figs. 5.3b and 5.3c are similarly motivated: use a few more keys and reassign

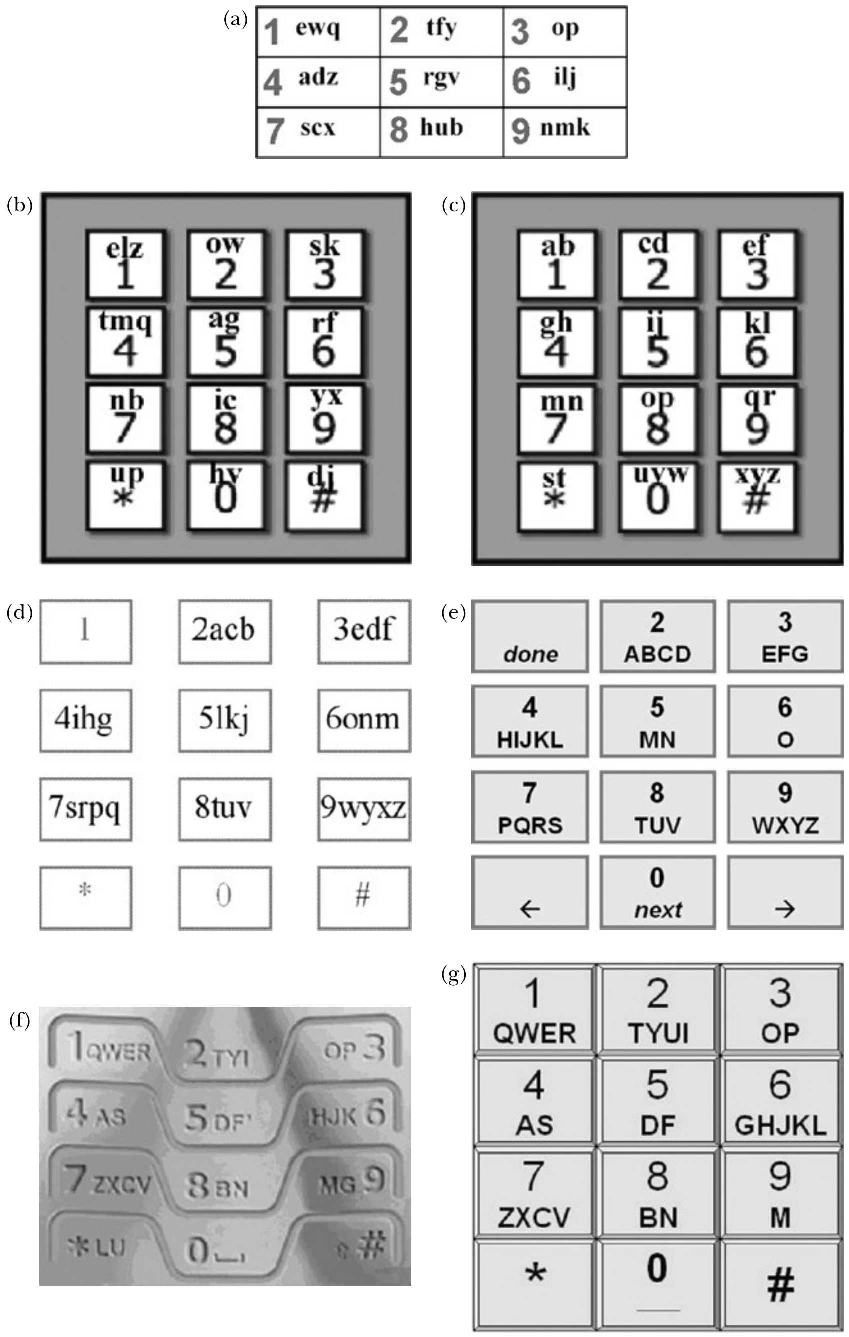

FIGURE 5.3 Mobile phone keypad variants. (a) Qwerty-like phone keypad (Hwang & Lee, 2005). (b) LetterEase (Ryu & Cruz, 2005). (c) FLpK (Ryu & Cruz, 2005). (d) LessTap (Pavlovych & Stuerzlinger, 2003). (e) ACD (Gong & Tarasewich, 2005). (f) EQ3 (eatoni.com). (g) QP10213.

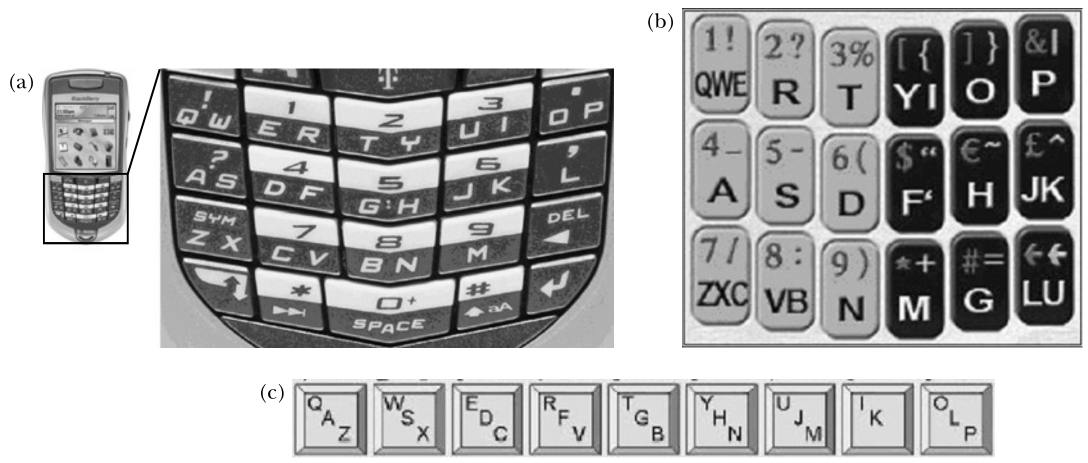

FIGURE 5.4 Qwerty keypad variants. (a) SureType (www.rim.com). (b) EQ6 (www.eatoni.com). (c) Stick (Green *et al.*, 2004).

letters to improve disambiguation. LetterEase in Fig. 5.3b does not use a familiar letter arrangement, whereas the fewer-letters-per-key (FLpK) design in Fig. 5.3c uses an alphabetical assignment. As with the Qwerty-like design of Hwang and Lee, it is thought that the familiar letter assignment will reduce the time to visually scan the layout in search of intended letters.

LessTap in Fig. 5.3d maintains the key and letter groupings of the phone keypad, while rearranging letters within keys. The goal, again, is to improve disambiguation. Gong and Tarasewich's (2005) alphabetically constrained design (ACD) in Fig. 5.3e uses the same 8 keys as a phone keypad, but with a strict alphabetic assignment. EQ3 by Eatoni (www.eatoni.com) in Fig. 5.3f uses 10 keys with a Qwerty-like assignment of letters. First author MacKenzie proposes the QP (Qwerty phone) design in Fig. 5.3g. It strictly adheres to the three-row Qwerty letter assignment, while using the phone keypad's top three rows of keys. The thought, again, is for a reduced visual scan time. For this design, an exhaustive search was performed using every possible assignment of letters, within these constraints. Of the 15,120 possibilities, enumeration 10,213 produces the lowest KSPC, hence the somewhat uninspired name of QP10213 for the design.

Figure 5.4 gives three interesting designs that do not utilize the three-column key arrangement in a phone keypad. Figure 5.4a is RIM's SureType keyboard (www.rim.com). Letters are assigned on 14 keys in a Qwerty arrangement. Figure 5.4b is Eatoni's EQ6, which assigns letters to 18 keys in a Qwerty-like arrangement. The G, L, and U letters are out of sequence to improve disambiguation. Figure 5.4c is the Stick of Green *et al.* (2004), which places letters on 9 keys in a single row. The P key is the lone outlier from the standard Qwerty arrangement.

One interesting research direction is to reduce the key complement below that of a phone keypad. Some examples are shown in Fig. 5.5. The TouchMe4Key of Tanaka-Ishii *et al.* (2002) in Fig. 5.5a is a predictive text entry system placing letters on four keys in alphabetic order within keys. The layout in Fig. 5.5b is the AKKO system for the motor impaired (Harbusch & Kuhn, 2003). The letter assignments were selected using a genetic algorithm that optimized the length of candidate word lists based on a built-in dictionary. The designs in Figs. 5.5c through 5.5f are some example arrangements assigning letters alphabetically to one, two, four, or six keys, respectively. They are denoted LnK, for letters on "*n*" keys.

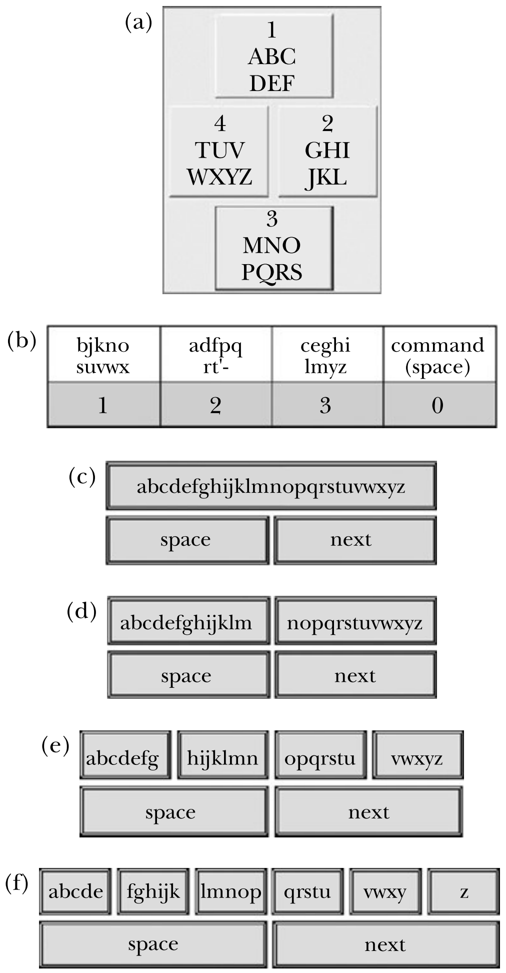

FIGURE 5.5 Fewer key keypad variants. (a) TouchMeKey4 (Tanaka-Ishii *et al.*, 2002). (b) AKKO (Harbusch & Kuhn, 2003). (c) Letters on one key. (d) L2K. (e) L4K. (f) L6K.

5.5 EVALUATING KEYBOARDS

The 17 designs in Figs. 5.2 through 5.5 are brought together for comparison in Table 5.1 along with the standard Qwerty keyboard as a point of reference. The entries are ordered by *T factor*, which is 1 plus the number of keys bearing letters. “One plus” reflects the need to include an additional key for SPACE, which constitutes about 18% of text entry in English. We use this convention to maintain consistency with Tegic’s *T9*—“text on nine keys,” which includes eight keys for letters (see Fig. 5.2) and a ninth key for SPACE. So, *T* factors range from 2 (Fig. 5.5c) to 27 for Qwerty. The columns Design and Figure refer to the discussions and figures above. The column Letter assignment notes, in a rough sense, the type of letter-to-key arrangement in each design. The last four columns show the statistics explained in Section 5.3. Two KSPC statistics are given for each entry method discussed earlier: one for multitap and the other for one-key with disambiguation. The last two columns show the degree of ambiguity when one-key with disambiguation is adopted.

Globally, there is a trade-off between the *T* factor and KSPC. This holds for both the non-predictive and the predictive cases. Figure 5.6 is offered as a visual improvement over the *T* factor and KSPC columns in Table 5.1. Here we clearly see a trade-off

				KSPC		**Degree of ambiguity**	
Design	**Figure**	**Letter arrangement**	***T* factor**	**Multitap**	**One-key**	**Average ranking K_{scan}**	**Conditional entropy**
L1K	5c	na	2	10.36647	20.05379	104.40538	6.36152
L2K	5d	ABC	3	5.14570	1.54712	3.96924	2.42934
AKKO	5b	Optimized	4	3.96958	1.09967	1.54090	1.05670
TMK4	5a	ABC	5	3.10980	1.05118	1.27776	0.63402
L4K	5e	ABC	5	3.52515	1.06697	1.36346	0.74735
L6K	5f	ABC	7	3.13954	1.02884	1.17902	0.43003
Phone	2	ABC	9	2.02422	1.00641	1.03479	0.11878
LessTap	3d	Optimized	9	1.49148	1.00641	1.03479	0.11878
ACD	3e	ABC	9	1.84078	1.00575	1.03122	0.09679
QP10213	3g	Qwerty	10	2.09677	1.00431	1.02337	0.08656
Stick	4c	Qwerty-like	10	1.59582	1.00545	1.05844	0.19253
QLPK	3a	Qwerty-like	10	1.32646	1.00537	1.03396	0.10037
EQ3	3f	Qwerty-like	11	1.84373	1.00227	1.01231	0.04446
FLpK	3c	ABC	13	1.43832	1.00597	1.03242	0.10547
LetterEase	3b	Optimized	13	1.21377	1.00265	1.01466	0.05514
SureType	4a	Qwerty	15	1.40525	1.00195	1.01059	0.04204
EQ6	4b	Qwerty-like	19	1.39502	1.00010	1.01701	0.06665
Qwerty	—	Qwerty	27	1.00000	1.00000	1.00000	0.00000

TABLE 5.1 KSPC and ambiguity for various keyboard designs and letter arrangements.

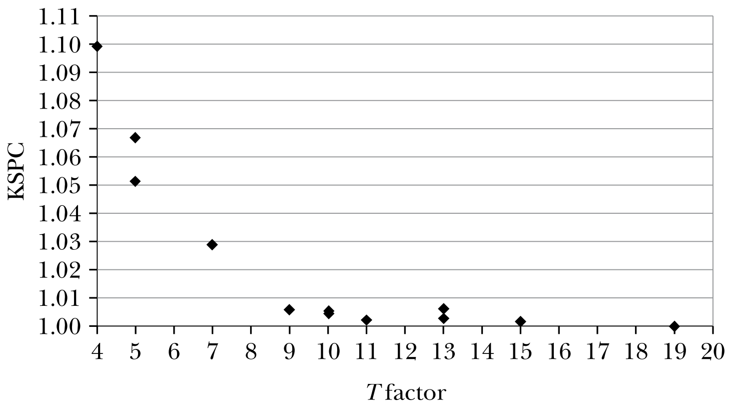

FIGURE 5.6 Trade-off between KSPC and *T* factor using data from Table 5.1. KSPC statistics are for one-key with disambiguation text entry.

in researchers' efforts to reduce the number of keys (*T* factor) and the ensuing KSPC. To show this relationship clearly, the L1K, L2K, and Qwerty entries are omitted in the chart. Despite this, KSPC figures are so close as to appear superimposed for *T* factors 9 (three entries) and 10 (three entries).

Another observation from Table 5.1 is that predictive methods reduce keystrokes. This is seen by comparing the columns under KSPC, one for multi-tap and the other for one-key with disambiguation. We see that the reduction in K_{code} is superior to the increase required for K_{scan} unless ambiguity is extreme, as in the L1K case.

As most points in Fig. 5.6 are close to 1.00, we offer an example of the impact of the ambiguity and the need to occasionally press NEXT to obtain the correct word. If a keyboard design has, say, KSPC = 1.012, the result is an additional 12 keystrokes per thousand. It does not sound like much, but let us see. In English, the average word size is 4.5 characters, or 5.5 characters counting spaces. So, the overhead is 1 additional keystroke for every 1000/5.5/12 = 15 words, such keystroke serving to select an alternative word from an ambiguous set. From a performance perspective, the problem is that users do not know when ambiguity will occur, except for common words or words frequently entered. As an input strategy, then, users typically attend to the display at the end of each word to determine if the correct word has appeared. And therein lurks the "attention demand" limitation of KSPC noted above.

Table 5.1 includes three measures concerning the degree of ambiguity: KSPC (for one-key with disambiguation), average ranking per word (K_{scan}), and conditional entropy. Figure 5.7 plots KSPC and average ranking for conditional entropy for each keyboard design. The horizontal axis shows the number of bits and the vertical axis shows the number of keystrokes. Both lines monotonically increase (almost linearly) as the conditional entropy is increased. We see here how the three scores globally

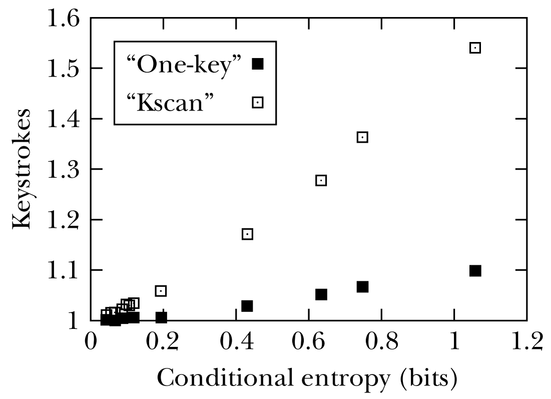

FIGURE 5.7 Correlation of KSPC and average ranking ambiguation text entry with conditional entropy.

correlate. Thus, KSPC and K_{scan} are globally governed by the complexity of candidates. Precisely speaking, though, KSPC in one-key with disambiguation includes K_{code}. The K_{code} component trades off with K_{scan} (see next section for a more explanation), so the conditional entropy sometimes is not exactly correlated with KSPC. In such cases, the conditional entropy, or K_{scan}, helps for a closer analysis.

Using such measures, the design is judged from the viewpoint of keystrokes. With multi-tap, there is no ambiguity and the user's actions are relatively uniform, so the KSPC number directly reflects the user's cognitive load. Therefore, among the keyboards with a T factor of 9 (for example), LessTap is the best. With the predictive entry methods, the keyboard should be judged by KSPC and the other measures directly showing the degree of ambiguity, although globally KSPC provides a good approximation. When the T factor is 9, we see that ACD is the best, which differs from the multitap case.

5.6 ENTRY BY COMPLETION

Entry by completion applies to either multi-tap or one-key with disambiguation. With multi-tap, entry changes from non-predictive to predictive: the system presents a list of candidates and the user selects from among them. The interesting design point is the keystroke reduction. This is an important benefit for users who enter text on mobile devices or for the disabled with reduced motor facility.

In this section, we discuss completion with the mobile phone keypad (summarized in the seventh row in Table 5.1, labeled Phone), for which the entry method is one-key

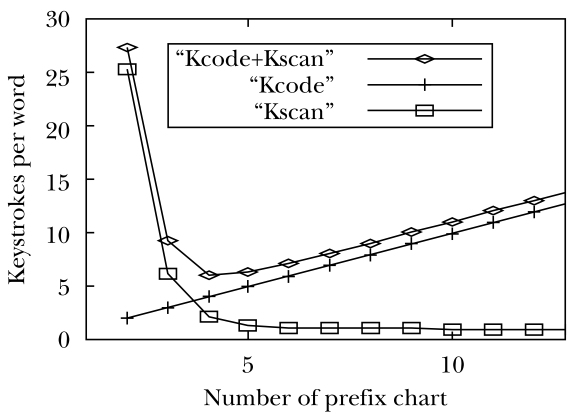

FIGURE 5.8 Minimum points for the number of keystrokes on a phone keypad.

with disambiguation with completion. The same discussion applies to entry methods on other keypads with completion, either multi-tap or one-key with disambiguation. As mentioned, a commercial version of such an entry system is manufactured by ZI Corp. Many entry systems for Japanese and Chinese also use this entry method.

Entry by completion is realized by entering the prefix and then selecting from among the candidates. Keystrokes are reduced by combining these two actions. Importantly, there is a trade-off between these two actions in terms of candidate complexity. The more of the prefix entered, the smaller the candidate list becomes, thus making it easier for the user to make a selection.

Figure 5.8 illustrates this for a ZI-like case in English. The x axis shows the length of the prefix (K_{code}) of the word and the y axis shows the keystrokes per word. To show the relationship clearly, the case with $K_{\text{code}} = 1$ is omitted in the chart. There are three lines in the figure. One plots $y = x$ ("+" markers), indicating the increasing value of K_{code}. Another (square markers) indicates the average ranking of the candidate for a given prefix length (K_{scan}). Clearly, the greater K_{code} is, the smaller K_{scan} becomes. The third (diamond markers) represents the keystrokes KSPW, the total of K_{code} and K_{scan}, with a minimum as a trade-off between K_{code} and K_{scan}. This line asymptotically approaches one of the two other lines at each end.

With a ZI-like keypad, the number of keystrokes is minimal at $K_{\text{code}} = 4$, where $K_{\text{scan}} = 2.13$. As the average length of an English word is 4.5 characters (excluding spaces), this means the user should enter almost the entire word to maximize the reduction in the number of keystrokes. For any other entry method using completion, the same discussion holds. For example, for Qwerty, the minimal point is at $K_{\text{code}} = 4$ (with $K_{\text{scan}} = 1.535$ to choose the candidate), and for TMK4 it is at $K_{\text{code}} = 5$ (with $K_{\text{scan}} = 2.619$). Curiously, for Qwerty, the minimal point is at the same word-beginning length as for the mobile phone, indicating that the degree of ambiguity is still large for cases with limited word-beginning length.

Another analysis is to compare methods with the same T factor. As shown in Table 5.1, ACD has less candidate complexity than Phone. However, ACD's minimal point is $K_{\text{code}} = 4$ with $K_{\text{scan}} = 2.23$, larger than that of a phone ($K_{\text{scan}} = 2.13$). Thus, a phone keypad is better if completion is used. Consequently, the degree of ambiguity when using completion is useful as a design aid.

In practice, the situation is more complicated. The above explanation assumes that the word-beginning length is predefined at a fixed value. However, the user may change the length while watching the candidates presented by the software. Such a decision affects the number of candidates shown at a time. These complicating factors are more finely modeled and measured by MacKenzie (2002), in which the minimal point is calculated using KSPC for list sizes of 1, 2, 5, and 10 on a Qwerty keyboard. These device options should be taken into consideration in exploring design scenarios.

5.7 SUMMARY AND FURTHER READING

Different keyboards can be designed for different tasks. Even though commercial de facto standard keyboards have predominated in the past, keyboard efficiency has become more important with the widespread adoption of ambiguous keyboards. We have explained ways to measure entry efficiency and expect the application of these techniques to lead to a better keyboard design.

For related work on ambiguous keyboards, the reader is directed to several sources, including Dunlop's five-key watch-top text entry system (Dunlop, 2004), Wobbrock and Myers' four-key gesture-based system (Wobbrock & Myers, 2006), and several eight- and nine-key designs described by Arnott and Javed (1992) and Lesher *et al.* (1998).

REFERENCES

Arnott, L. J., & Javed, Y. M. (1992). Probabilistic character disambiguation for reduced keyboards using small text samples. *Augmentative and Alternative Communications, 8,* 215–223.

Bell, T., Cleary, J., & Witten, I. H. (1990). *Text compression.* Upper Saddle River, NJ: Prentice Hall.

Darragh, J. J., Witten, I. H., & Long, J. (1992). *The reactive keyboard.* Cambridge, UK: Cambridge University Press.

Desautels, E. J., & Soffer, S. B. (1974). Touch-tone input techniques: Data entry using a constrained keyboard. *Proceedings of the 1974 Annual Conference* (pp.245–253). New York: ACM Press.

Dix, A., Finlay, J., Abowd, G., & Beale, R. (2004). *Human–computer interaction (3rd ed.).* London: Prentice Hall.

Dunlop, M. (2004). Watch-top text-entry: Can phone-style predictive text entry work with only 5 buttons? *Proceedings of Mobile HCI 2004* (pp.342–346). Heidelberg: Springer-Verlag.

Gong, J., & Tarasewich, P. (2005). Alphabetically constrained keypad designs for text entry on mobile phones. *Proceedings of the ACM Conference on Human Factors in Computing Systems—CHI 2005* (pp.211–220). New York: ACM Press.

Green, N., Kruger, J., Faldu, C., & Amant, R. S. (2004). A reduced QWERTY keyboard for mobile text entry. *Extended Abstracts of the ACM Conference on Human Factors in Computing Systems—CHI 2004* (pp.1429–1432). New York: ACM Press.

Harbusch, K., & Kuhn, M. (2003). Towards an adaptive communication aid with text input from ambiguous keyboards. *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics—EACL 2003.* Cambridge, MA: MIT Press.

Higginbotham, D. J. (1992). Evaluation of keystroke savings across five assistive communication technologies. *Augmentative and Alternative Communications, 8,* 258–272.

Hwang, S., & Lee, G. (2005). Qwerty-like 3 × 4 keypad layouts for mobile phone. *Extended Abstracts of the ACM Conference on Human Factors in Computing Systems—CHI 2005* (pp.1479–1482). New York: ACM Press.

Koester, H. H., & Levine, S. P. (1998). Model simulation of user performance with word prediction. *Augmentative and Alternative Communications, 14,* 25–35.

Kurihara, T. (1970). Kana-Kanji Conversion (I). *Technology reports of Kyushu University, 42(6),* 880–884, in Japanese. Hakata: Kyushu University.

Kurihara, T. & Kurosaki, Y. (1967). On the Transformation Process of Phonetic Sentences into Ideographic Sentences. *Technology reports of Kyushu University, 39(4),* 659–664, in Japanese. Hakata: Kyushu University.

Lesher, G. W., Moulton, B. J., & Higginbotham, D. J. (1998). Optimal character arrangements for ambiguous keyboards. *IEEE Transactions on Rehabilitation Engineering, 6,* 415–423.

MacKenzie, I. S. (2002). KSPC (keystrokes per character) as a characteristic of text entry techniques. *Proceedings of the Fourth International Symposium on Human–Computer Interaction with Mobile Devices* (pp.195–210). Heidelberg: Springer-Verlag.

Pavlovych, A., & Stuerzlinger, W. (2003). Less-Tap: A fast and easy-to-learn text input technique for phones. *Proceedings of Graphics Interface 2003* (pp.97–104). Toronto: Canadian Information Processing Society.

Rabiner, L. R., & Schafer, R. W. (1976). Digital techniques for computer voice response: Implementations and applications. *Proceedings of the IEEE, 64,* 416–433.

Ryu, H., & Cruz, K. (2005). LetterEase: Improving text entry on a handheld device via letter reassignment. *Proceedings of the 19th Conference of the Computer–Human Interaction Special Interaction Group (CHISIG) of Australia* (pp.1–10). New York: ACM Press.

Shannon, C. E. (1951). Prediction and entropy of printed English. *Bell System Technical Journal, 30,* 51–64.

Silfverberg, M., MacKenzie, I. S., & Korhonen, P. (2000). Predicting text entry speed on mobile phones. *Proceedings of the ACM Conference on Human Factors in Computing Systems—CHI 2000* (pp.9–16). New York: ACM Press.

Smith, S. L., & Goodwin, N. C. (1971). Alphabetic data entry via the touch tone pad: A comment. *Human Factors, 13*, 189–190.

Tanaka-Ishii, K., Inutsuka, Y., & Takeichi, M. (2002). Entering text with a four-button device. *Proceedings of the 19th International Conference on Computational Linguistics, Vol. 1* (pp.1–7). Morristown, NJ: Association for Computational Linguistics.

Wobbrock, J. O., & Myers, B. A. (2006). Few-key text entry revisited: Mnemonic gestures on four keys. *Proceedings of the ACM Conference on Human Factors in Computing Systems—CHI 2006* (pp.489–492). New York: ACM Press.