HUMAN-COMPUTER INTERACTION, 2002, Volume 17, pp. 147–198
Copyright © 2002, Lawrence Erlbaum Associates, Inc.

Text Entry for Mobile Computing: Models and Methods, Theory and Practice

I. Scott MacKenzie and R. William Soukoreff
York University

ABSTRACT

Text input for mobile or handheld devices is a flourishing research area. This article begins with a brief history of the emergence and impact of mobile computers and mobile communications devices. Key factors in conducting sound evaluations of new technologies for mobile text entry are presented, including methodology and experiment design. Important factors to consider are identified and elaborated, such as focus of attention, text creation versus text copy tasks, novice versus expert performance, quantitative versus qualitative measures, and the speed–accuracy trade-off. An exciting area within mobile text entry is the combined use of Fitts' law and a language corpus to model, and subsequently optimize, a text entry technique. The model is described, along with examples for a variety of soft keyboards as well as the telephone keypad. A survey of mobile text entry techniques, both in research papers and in commercial products, is presented.

I. Scott MacKenzie is a computer scientist with an interest in human–computer interaction, especially human input to computing systems and human performance measurement and modeling; he is an Associate Professor in the Department of Computer Science at York University. **R. William Soukoreff** is a computer scientist with an interest in human–computer interaction; he is a graduate student in the Department of Computer Science at York University.

CONTENTS

1. INTRODUCTION

Although text entry is by no means new in mobile computing, there has been a burst of research on the topic in recent years. There are several reasons for this heightened interest: First, mobile computing is on the rise and has spawned new application domains such as wearable computing, two-way paging, and mobile Web and e-mail access. Second, word processors, spreadsheets, personal schedulers, and other traditional desktop applications are increasingly available on mobile platforms. Third, there is a strong demand

for the input of text or alphanumeric information that is easily and efficiently entered, recognized, stored, forwarded, or searched, via traditional software techniques. Fourth, the phenomenal success of text messaging with mobile phone users has inspired considerable speculation on future spin-off technologies, all expected to benefit from text entry.

The statistics for text messaging on mobile phones are remarkable. In January 2001, GSM Europe reported that 15 billion Short Message Service (SMS) text messages are transmitted per month worldwide.[1] This is particularly interesting in view of the limited capability for text input with the current generation of mobile phone technology.

Although the ubiquitous Qwerty keyboard reigns supreme as the primary text entry device on desktop systems, mobile and handheld systems lack an equivalent dominant technology or technique for the same task. And so, the challenge of text entry for mobile computing presents itself. A valid question is, Why not just apply the Qwerty keyboard to the mobile paradigm? Despite the obvious advantage of familiarity, a Qwerty keyboard is bulky, and unless the keyboard is full size, touch typing is hampered or impossible. In addition, some mobile devices are intended for one hand use, and this reduces the advantage of the Qwerty arrangement. (A notable exception is the *Half-Qwerty* keyboard, discussed later.) Many mobile devices are committed to the pen input paradigm, so a Qwerty keyboard is simply not an option. Where physical buttons or keys are employed, the mobile form factor often limits the key complement to a dozen or fewer keys.

This article is organized as follows. We begin with a brief historical background of mobile and handheld devices. This is important because it juxtaposes the efforts of researchers with the corporations that created early mobile and handheld devices. Following this, we offer some opinions and observations on the evaluation of text input techniques. Many, but not all, of the techniques described later in this article have been empirically evaluated in user tests. To compare input technologies, the results of these evaluations are crucial. Factors to consider are presented and elaborated. Following this, we detail one of the most active areas of current research—optimization of text entry using language and motor control modeling. Finally, we present a survey of the current state of the art in text entry for mobile computing. We conclude with some observations on the technologies reviewed and the open research questions that remain.

---

1. GSM stands for Global System for Mobile communications. The GSM Association, based in Dublin, Ireland, represents the interests of hundreds of satellite operators, manufacturers, suppliers, and regulatory and administrative bodies from around the world. See http://www.gsmworld.com for further details.

There are two notable omissions in this article. One is speech recognition as a vehicle for text entry. Always "about to emerge," speech is an input technology quick to grab headlines but perennially unable to enter the mainstream of computing. In our view, speech is a deserving (albeit niche) technology, but it is unlikely to supplant traditional interaction techniques for desktop or mobile computing (see Shneiderman, 2000, for further discussion). Although some mobile phones support limited speech recognition, the interaction consists of selecting from a small list of preprogrammed entries, such as names in an address book. Speech recognition is not used for general purpose text input on mobile devices.

The other omission is international languages. It is clear and obvious that text entry does not imply "English text entry." Languages throughout the world are currently supported in various forms in mobile computing, and this will continue. Although the focus in this article is on English, the discussions apply to other languages, particularly those based on the Roman alphabet (see Sacher, 1998, for a discussion on text entry in Asian languages).

1.1. Mobile Computing

Among the earliest of handheld devices was the HP95LX, which was released in 1991 by Hewlett-Packard (Palo Alto, CA; http://www.hp.com). The technological equivalent of an IBM® XT shrunk into a clamshell format, the HP95LX was small enough to fit in the palm of one's hand. Although the term *Personal Digital Assistant* (PDA) had not yet entered the vernacular to describe a handheld computer, this was the first PDA. The HP95LX provided a small Qwerty keyboard for text entry, although touch typing was impossible due to its size. Later devices (the HP100LX and HP200LX) followed. These devices demonstrated that the Qwerty keyboard could be adapted to mobile computing devices.

The early 1990s was an exciting time for mobile computing due to the arrival of pen computing. The ideas touted much earlier by Kay and Goldberg (1977) in their Dynabook project finally surfaced in commercial products. However, the initial devices were bulky, expensive, and power-hungry, and they could not deliver in the one area that garnered the most attention—handwriting recognition. Without a keyboard, the pen was the primary input device. If only "selecting" and "annotating" were required, then the success of pen entry seemed assured. However, some applications demanded entry of text as machine-readable characters, and the handwriting recognition technology of the time was not up to the challenge. Products from this era, such as GRidPaD, Momenta™, Poqet®, and PenPad, did not sustain the volume of sales necessary for commercial viability. Most endured only 1 or 2 years.

One of the most significant events in pen-based computing was the 1993 announcement from Apple® Computer, Inc. (Santa Clara, CA; http://www.apple.com) that it would enter the pen computing market. Thus emerged the Apple MessagePad® (a.k.a. Newton®). Apple was a major player in desktop computing at the time, and its commitment to pen computing was taken seriously. To a certain extent, Apple added legitimacy to this entire segment of the computing market. However, the Newton was expensive and rather specialized. It was embraced by many technophiles, but it did not significantly penetrate the larger desktop or consumer market. The Newton's handwriting recognition, particularly on early models, was so poor that it was ridiculed in the media—by Garry Trudeau (1996), for example, in his celebrated *Doonesbury* cartoons. Nevertheless, the Newton received considerable attention, and it ultimately set the stage for future mobile devices.

The next significant event in mobile or pen computing was the release in 1996 of the Palm™ Pilot (now called the Palm) by Palm Inc. (Santa Clara, CA; http://www.palm.com). The Palm was an instant hit. Five years hence, it is the technology of choice for millions of users of mobile devices. There is much speculation on why the Palm was so successful. Some factors seem relevant: The price was about $500, a few hundred less than a Newton. The Palm supported HotSync® (including cables and software for transferring data between the Palm and a desktop computer) as a standard feature. The Palm was smaller and lighter than the Newton, and could fit in one's pocket. Because of lower power consumption, the batteries lasted for weeks instead of hours. Finally, and perhaps most important, the Palm avoided the thorny issue of cursive or block-letter handwriting recognition by introducing a greatly simplified handwriting technique known as Graffiti® (which is discussed later in this article). By simplifying recognition, Graffiti required less CPU power and memory, achieved better character recognition, and ultimately enjoyed widespread acceptance among users.

The year 1996 also saw the release of the Windows® CE operating system by Microsoft® (Redmond, WA; http://www.microsoft.com). Devices such a Casio's Cassiopeia® or Philips' Velo, which used Windows CE, were more powerful than previous mobile computing devices, but were also larger. The first version of Windows CE only supported a soft keyboard for text entry, but later versions included the JOT handwriting recognizer, by Communications Intelligence Corporation® (Redwood Shores, CA; http://www.cic.com), and Microsoft Transcriber.

A recent entry in the pen computing market is the CrossPad by A. T. Cross Company (Lincoln, RI; http://www.cross.com). The CrossPad avoids handwriting recognition by recording the user's writing as ink trails. The user's notes are downloaded to a desktop computer for storage and subsequent recognition on the desktop computer. Software accompanying the CrossPad sup-

ports handwriting recognition of keywords, and indexing and retrieval by keyword.

All of the devices previously mentioned are handheld computers that support text entry. Another quite different group of devices that support text entry are messaging devices such as mobile phones and pagers. In Europe, where text messaging has been available since 1991, more text messages are transmitted daily than voice messages.[2] In North America, most mobile phones and pagers do not yet support text messaging, but this is changing. The latest generation of two-way pagers such as the BlackBerry™ by Research In Motion (Waterloo, Ontario, Canada; http://rim.net, but also see http://www.blackberry.net) and PageWriter® by Motorola (Schaumburg, IL; http://www.motorola.com) support text entry via a miniature Qwerty keyboard.

We conclude this perspective with a hint at what the future might hold. At the top of our list is a device combining the programmability of the PDA, wireless telephony, text messaging, and unfettered Internet and e-mail access. Pieces of this scenario already exist, but implementations require a specialized configuration, optional components, or support only a subset of standard features. We view these as transitional technologies. Devices that do not quite make the grade, in our view, are those that require an add-on radio transceiver or provide Internet access only to sites supporting a specialized protocol (e.g., wireless access protocol).

For text input, the pen-based paradigm has dominated the PDA market, but there is a parallel trend toward text messaging in mobile phones and pagers using keyboard-based technology. If these technologies converge, then which text input technology will prevail? This is a difficult question to answer, and although there is no definitive answer, the following section identifies the key issues to consider.

1.2. Text Entry

There are two competing paradigms for mobile text input: pen-based input and keyboard-based input. Both emerged from ancient technologies ("ancient" in that they predate computers): typing and handwriting. User experience with typing and handwriting greatly influences expectations for text entry in mobile computing; however, the two tasks are fundamentally different.

---

2. SMS (Short Message Service) is the predominant text-messaging technology in Europe. SMS supports transmission and reception of messages up to 160 characters via mobile phones (phone-to-phone). *Instant Messaging* is a similar technology popular in North America, but it is used mostly in PC-to-phone messaging.

A key feature of keyboard-based text entry is that it directly produces machine-readable text (i.e., ASCII characters), a necessary feature for indexing, searching, and handling by contemporary character-based technology. Handwriting without character recognition produces "digital ink." This is fine for some applications such as annotation, visual art, and graphic design. However, digital ink requires more memory and in general it is not well managed by computing technology. Specifically, digital ink is difficult to index and search (although Poon, Weber, & Cass, 1995, reported some success with a graphical search mechanism for digital ink that is not based on recognition). For handwritten text entry to achieve wide appeal, it must be coupled with recognition technology.

An important consideration implicit in the discussion of text input technology is user satisfaction. The point was made earlier that the Palm succeeded where the Newton failed, in part because of users' acceptance of Graffiti as a text input technology. Users' expectations for text entry are set by current practice. Modest touch typing speeds in the range of 20 to 40 words per minute (wpm) are achievable for hunt-and-peck typists. Rates in the 40 to 60 wpm range are achievable for touch typists, and with practice, skilled touch typists can achieve rates greater than 60 wpm. Handwriting speeds are commonly in the 15 to 25 wpm range. These statistics are confirmed by several sources (Card, Moran, & Newell, 1983; Devoe, 1967; Lewis, 1999; MacKenzie, Nonnecke, Riddersma, McQueen, & Meltz, 1994; Van Cott & Kinkade, 1972). Users, perhaps unrealistically, expect to achieve text input rates within these ranges on mobile devices. Furthermore, they expect these rates immediately, or within a short time of using a new input technology.

The preceding paragraphs have outlined qualities of a successful text input method. Production of machine-readable characters at a speed acceptable to users is a reasonable objective. To determine if a particular text input method meets this objective, or to compare new and existing text input methods, a user evaluation is needed.

2. EVALUATION

Research in mobile text entry is flourishing in part because user needs are not currently met. Typically, traditional text input technologies are refined or new input technologies are invented. Either way, evaluation is a critical and demanding part of the research program. The questions researchers pose are ambitious: Can entry rates be improved if we arrange the buttons on a keyboard in a certain way? What is the effect if we use context to guess the next letter or word? Can we apply an altogether different technology, like pie menus, touch pads, or pattern recognition, to the problem of text input? In this section,

we discuss important issues in undertaking valid and useful evaluations of text entry techniques.

2.1. Methodology

An evaluation is valuable and useful if the methodology is reproducible and results are generalizable. *Reproducible* implies that other researchers can duplicate the method to confirm or refute results. This is achieved for the most part simply by following an appropriate reporting style (e.g., American Psychological Association, 1995). *Generalizable* implies that results have implications beyond the narrow context of the controlled experiment. This is achieved through a well-designed experiment that gathers measures that are accurate and relevant, in tasks that are representative of real-life behavior. There is, unfortunately, a trade-off here. In real life, people rarely focus solely on a single task. Methodologies so designed, therefore, may find that the measurements include behaviors not specifically required of the interaction technique. The trade-off, therefore, is between the accuracy of our answers and the importance or relevance of the questions they seek to address. That is, we can choose between providing *accurate answers* to *narrow questions*, or providing *vague answers* to *broad questions*. The reader is implored not to interpret this too strictly, but, we hope, the point is made. In designing an experiment, we strive for the best of both worlds; answering interesting or broad questions (viz., using real-life tasks) and doing so accurately (viz., accurately measuring the behavior of interest, such as entry speed or accuracy).

In the following sections, we identify some factors relevant to methodologies for evaluating text entry on mobile systems.

2.2. Text Creation Versus Text Copy

An important distinction in text entry evaluations is between *text creation* and *text copy*. In a text copy task, the participant is given text to enter using the input technique under investigation. In a creation task, the source text is either memorized or generated by the participant. Although text creation is closer to typical usage, the approach is generally not appropriate for an empirical evaluation. This is explained in the following paragraphs.

As a backdrop for discussing these two types of tasks, we introduce the term *focus of attention* (FOA). FOA speaks to the attention demands of the task. Consider the case of an expert touch typist using a Qwerty keyboard to copy text from a nearby sheet of paper. This is a text copy task. Because the typist is an expert, she does look at the keyboard or display—she attends only to the source text. This is a single FOA task. However, if input is via a stylus and soft keyboard, the typist must also attend to the keyboard. (A soft keyboard cannot

be operated "eyes free.") Stylus typing, therefore, is a two-FOA task. If the typist is at a less-than-expert level and corrects errors, she must look at the display to monitor results. This increases touch typing to a two-FOA task and stylus typing to a three-FOA task. Clearly, the feedback channel is overburdened in a three-FOA task.

Despite the additional FOA, text copy tasks are generally preferred to text creation tasks for empirical evaluations. There are several reasons. One is the possible presence of behaviors not required of the interaction technique. Examples include pondering ("What should I enter next?") or secondary tasks (e.g., fiddling with system features). Clearly, measurement of text entry speed is compromised if such behaviors are present.

A second difficulty with text creation tasks is identifying errors—it is difficult to know exactly what a participant intended to enter if the participant is generating the text. Even if the message content is known a priori, errors in spelling or memory recall may occur, and these meta-level mistakes are often indistinguishable from errors due to the interface itself.

A third difficulty is the loss of control over the distribution of letters and words entered. The task should require the participant to enter a representative number of occurrences of characters or words in the language (i.e., results are generalizable). However, it is not possible to control for this if the participant is generating the text.

The main advantage of a text creation task is that it mimics typical usage. The disadvantages just cited, however, are significant and drive most researchers to use text copy tasks despite the increased FOA. One way to mitigate the effects of increased FOA is to dictate the source text through the audio channel. Ward, Blackwell, and MacKay (2000) used this technique; however, they noted that participants found the approach stressful and hard to follow.

A carefully designed experiment may capture the strengths of both a text creation task and a text copy task. One technique is to present participants with short, easy-to-memorize phrases of text. Participants are directed to read and memorize each phrase before entering it. Entry proceeding thus benefits from the desirable property of a text creation task (viz., reduced FOA). In addition, the desirable properties of a text copy task are captured (i.e., control over letter and word frequencies and performance measurements that exclude thinking about what to write). There are numerous examples of this approach in the literature (e.g., Alsio & Goldstein, 2000; MacKenzie, Nonnecke, Riddersma, et al., 1994; MacKenzie & Zhang, 1999; Rau & Skiena, 1994). A similar technique is to present text in large a block (e.g., a complete paragraph) but to interleave each line of the presented text (input) with each line of generated text (output). As input proceeds, each character entered appears directly below the intended character. This is a text copy task; however, FOA is reduced to that of a text creation task because participants attend only to one lo-

cation for both the source text and the results of entry. An example of this methodology is reported by Matias, MacKenzie, and Buxton (1993) and Matias, MacKenzie, and Buxton (1996a).

2.3. Novice Versus Expert Performance

Most work on the design of text input methods focuses on the potential, or expert, text entry rate of a particular design. However, the novice experience is paramount for the success of new text input methods. This is at least partially due to the target market. Mobile devices, such as mobile phones and PDAs, once specialized tools for professionals, are increasingly targeted for the consumer market. It follows that "immediate usability" is important. In other words, it may be a moot point to establish the expert, or "potential" text entry rate for an input technique if prolonged practice is required to achieve it. Consumers, discouraged by their initial experience and frustration, may never invest the required effort to become experts.

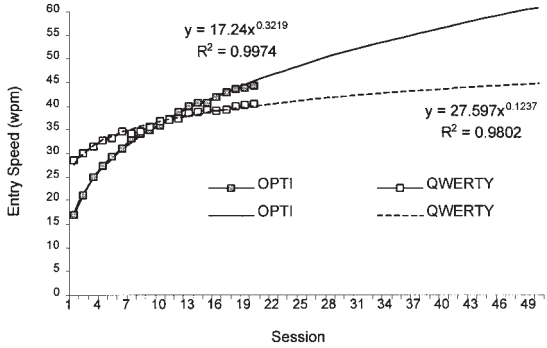
However, measuring immediate usability is easier said than done. In typical studies of new interaction techniques, participants are given a demonstration of the technique followed by a brief practice session. Then, data collection proceeds over several blocks of trials. However, the measurements are a poor indicator of novice behavior, at least in the sense of immediate, or walk-up, usability. Within a few minutes, participants' knowledge of the interaction technique develops, and the novice status fades. Measuring expert performance is also not easy because acquisition of expertise requires many blocks of trials administered over many days, or more.

Some longitudinal text entry studies are hereby cited (Bellman & MacKenzie, 1998; Gopher & Raij, 1988; MacKenzie & Zhang, 1999; Matias et al., 1996a; McMulkin, 1992). An example of results from a typical longitudinal study is given in Figure 1. Users' improvement in entry speed is shown over 20 sessions of input for two types of soft keyboard. The data were fitted to the standard power law of learning (see Card, English, & Burr, 1978). Prediction equations and squared correlations are shown, as are extrapolations of the predictions to 50 sessions.

2.4. Quantitative Versus Qualitative Analyses

We noted earlier a trade-off between the accuracy of answers and the relevance of the questions they seek to address. Quantitative evaluations tend to provide accurate answers to narrow questions, whereas qualitative evaluations tend to provide rather loose answers ("participants liked the device!") to broad but very important issues (comfort, ease of use, subjective impression, etc.). Of course, researchers strive for the best of both worlds. In quantitative evalua-

Figure 1. **Reporting example for a longitudinal study. From "The design and evaluation of a high-performance soft keyboard," by I.S. MacKenzie and S. X. Zhang, 1999, *Proceedings of the CHI 99 Conference on Human Factors in Computing Systems.* Copyright 1999 by ACM. Reprinted with permission.**



tions, "representative tasks" and "relevant measures" are used to ensure interesting or relevant questions are answered. In qualitative evaluations, robust test instruments are developed to ensure the answers are accurate, relevant, reproducible, and generalizable.

When reporting quantitative results, there are many common pitfalls to avoid such as inaccuracy in measurements, lack of control or baseline conditions, inferring too much from data, using too small a sample size, collecting insufficient data, artificially biasing data by aggregation, nonrandom presentation of conditions, and inappropriate treatment of outliers. The reader is directed to textbooks in experimental psychology for further discussions (e.g., Martin, 1996; for a discussion on aggregation bias, see Walker et al., 1993).

Researchers may be excused for slightly bending the rules, perhaps, but all too common are published reports stating only qualitative results steeped in anecdote, or, worse yet, testimonials unsupported by empirical data. An excerpt from one such publication illustrates our point:

> While we have yet not done systematic user testing, anecdotal experience to date is consistent: Users well practiced in both … and … consistently find the latter to be about three times faster, with accuracy for both systems very high.[3]

Testimonials such as this are of questionable merit; they surely do not meet the criteria for good research—that results are generalizable and reproducible.

---

3. An excerpt from a paper published in the proceedings of a conference in human–computer interaction.

Unless a controlled experiment is performed using quantitative metrics or established qualitative test instruments, there is no way to gauge the performance of a new text input technique. Conjuring up a new input technique is fine, but research demands more. It demands that new ideas are implemented and evaluated in conformance with the rigors of an empirical evaluation.

Although quantitative tests form the backbone of any scientific study, qualitative aspects of the investigation are also important. In human–computer interfaces, users must feel comfortable with the interaction technique and must feel their efforts have a reasonable payoff in their ability to accomplish tasks. Participants will develop impressions of each device or condition tested, and these should be solicited and accounted for in the final analysis. Typically, these opinions are sought via questionnaire, administered at the end of a condition or experiment. The reader is referred to textbooks in human–computer interaction for direction in questionnaire design (e.g., Dix, Finlay, Abowd, & Beale, 1998).

2.5. Speed

For text input there are two primary evaluation metrics: speed and accuracy. The simplest way to measure and report speed is to measure the number of characters entered per second during a trial, perhaps averaged over blocks of trials. This gives a measure in characters per second (cps). To convert this to wpm, the standard typists' definition of a word as five characters (regardless of whether the characters are letters, punctuation, or spaces) is employed (Gentner, Grudin, Larochelle, Norman, & Rumelhart, 1983). Therefore, wpm is obtained by multiplying characters per second by 60 (seconds per minute) and dividing by 5 (characters per word).

2.6. Accuracy

Accuracy is more problematic. For a simple treatment of accuracy, we obtain a metric that captures the number of characters in error during a trial and report these as a percentage of all characters in the presented text. A more complete analysis involves determining what kind of errors occurred, and why. The difficulty arises from the compounding nature of mistakes (see Suhm, Myers, & Waibel, 1999), and the desire to automate as much of the data measurement and analysis as possible. Four basic types of errors include entering an incorrect character (substitution), omitting a character (omission), adding an extra character (insertion), or swapping neighboring characters (transposition). Although it is straightforward for a human to compare the intended text with the generated text and tabulate the errors, in practice the amount of analysis is simply too much, given a reasonable number of partici-

pants, conditions, and trials. Additionally, tabulation errors may be introduced if performed manually.

However, automating error tabulation is not trivial. Consider an experiment where the participant is required to enter the 19-character phrase "the quick brown fox." If the participant enters "the quxxi brown fox," the incorrect word contains either three substitution errors or two insertion ("xx") and two omission ("ck") errors. The explanation with the fewest number of errors (3) is preferred and, in this simple example, yields an error rate of (3 / 19) × 100% = 15.8%. Algorithms for "string distance" calculations, such as the Levenshtein string distance statistic (Damerau, 1964; Levenshtein, 1966), might assist in automating analyses such as these, as demonstrated by Soukoreff and MacKenzie (2001).

Difficulties in error tabulation have pushed some researchers to ignore errors altogether (e.g., Venolia & Neiberg, 1994) or to force the participant to enter correct text only (e.g., Lewis, 1999).

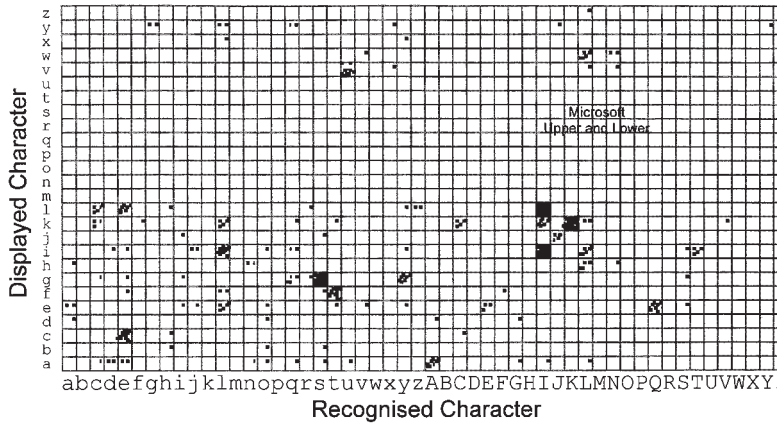
Directing participants to "correct as you go" is another possible approach. Assuming participants adhere to instructions, the resulting text is error free; thus, the error rate is 0%. In general though, participants will leave errors in the generated text, even if requested not to. The result is two levels of errors—those that were corrected and those that were not. For the corrected errors, overhead is incurred in making the corrections. A reasonable measure of accuracy in this case is keystrokes per character (KSPC). For a Qwerty keyboard, the ideal is KSPC = 1.0, but, in practice, KSPC > 1 if participants correct as they go. If, for example, a 25-character phrase were entered and two substitution errors occurred, each corrected by pressing Backspace followed by the correct character, then KSPC = (25 + 4) / 25 = 1.16.[4]

A useful tool for designers is the *confusion matrix,* graphically depicting the frequency of character-level transcription errors. Figure 2 is a confusion matrix taken from MacKenzie and Chang's (1999) comparative study of two handwriting recognizers. The confusion matrix displays intended characters versus recognized characters illustrating how often an intended character (left-hand column) was misrecognized and interpreted as another character (bottom row). Each dot represents three occurrences.

Clearly, both speed and accuracy must be measured and analyzed. Speed and accuracy are commonly known to exist in a continuum, wherein speed is traded for accuracy and vice versa (Hancock & Newell, 1985; Pachella & Pew, 1968; Pew, 1969; Swensson, 1972; Wickelgren, 1977). Participants can enter

4. Correct-as-you-go has an additional problem: reaction time. If entry proceeds quickly, an error may be followed by several additional entries before the participant can react to the error. The overhead in correcting the error may be substantial (see Matias, Mackenzie, & Buxton, 1996a, for a discussion of this).

Figure 2. **Sample confusion matrix. From "A performance comparison of two handwriting recognizers," by I. S. MacKenzie and L. Chang, 1999, *Interacting with Computers, 11,* pp. 283–297. Copyright 1999 by Elsevier Service. Reprinted with permission.**



text more quickly if they are willing to sacrifice accuracy. For participants to perform with high accuracy, they must slow down. The trade-off suggests that measuring only speed or only accuracy will skew the results so as to make the text input method appear better (or worse!) than it really is. An example of a reporting technique that combines speed and accuracy is given in Figure 3. Conditions are "better" toward the top and right of the figure because they are both fast and accurate.

2.7. Other Factors

The text input process can be significantly impacted by factors bearing little on the input device, such as whether the device is operated standing, sitting, or walking, or whether it is operated with one or two hands. Designers of novel text input techniques must be aware that users want to operate mobile devices anytime, anywhere. Lack of a one-hand interaction method may impact the commercial success of a technology.

Evaluations are often conducted to test a refinement to existing practice. Often the new technique is an improvement over the status quo. In the next section, we present some key initiatives in improving current practice through language and movement modeling.

3. OPTIMIZATION TECHNIQUES

There are two popular approaches to optimizing the text entry task: movement minimization and language prediction. Movement minimization seeks

Figure 3. **Simultaneous presentation of results for speed and accuracy. From "A comparison of three methods of character entry on pen-based computers," by I.S. MacKenzie, R. B. Nonnecke, J. C. McQueen, S. Riddersma, and M. Meltz, 1994, *Proceedings of the Human Factors Society 38th Annual Meeting.* Copyright 1994 by the Human Factors and Ergonomics Society. Reprinted with permission.**

Lawrence Erlbaum Associates, Inc. does not have electronic rights to Figure 3. Please see the print version.

to reduce the movements of the finger or pen in interacting with a mobile device to enter text. Language prediction exploits the statistical nature of a language to predict the user's intended letters or words. There are also hybrid approaches. In the following sections we summarize these modeling and design techniques.

3.1. Movement Minimization

The main reason for using a Qwerty keyboard for text input is to support touch typing. The next-best reason is familiarity with the letter arrangement. However, the sheer size of a Qwerty keyboard is imposing and ill-suited to the mobile paradigm. Recent work has focused on the limited case of single-finger or stylus entry, either on a soft keyboard or on a small physical keyboard with a reduced key set. This work combines a statistical language model with a movement time prediction model to assist in modeling and designing input techniques wherein device or hand movement is as efficient as possible (Hunter, Zhai, & Smith, 2000; Lewis, Allard, & Hudson, 1999a; Lewis, LaLomia, & Kennedy, 1999a, 1999b; MacKenzie & Zhang, 1999; MacKenzie, Zhang, & Soukoreff, 1999; Zhai, Hunter, & Smith, 2000; Zhang, 1998). The following is the summary of a model introduced by Soukoreff and MacKenzie (1995).

The model comprises five major components: (a) a digitized layout of a keyboard; (b) Fitts' law for rapid aimed movements; (c) the Hick–Hyman law for choice selection time; (d) a linguistic table for the relative frequencies of letter

pairs, or digrams, in common English; and (e) a spreadsheet or software tool in which the preceding components are combined.

For (a), each key is assigned an *x*–*y* coordinate, thus allowing digram distances to be easily computed using the Pythagorean identity. For (b), we use Fitts' law (Fitts, 1954; MacKenzie, 1992) to predict the movement time (*MT*; in seconds) to tap any key given any previous key. This is a simple prediction based on the distance between the keys (A_{ij}) and the size, or width, of the target key (W_j):

$$MT_{ij} = 0.204 \log_2 \left(\frac{A_{ij}}{W_j} + 1 \right) \tag{1}$$

For (c), we use the Hick–Hyman law (Hick, 1952; Hyman, 1953) to predict the reaction time (*RT*; in seconds) to visually scan a 27-key layout to find the target key. For novices, we set

$$RT = 0.200 \log_2 (27) = 0.951 \text{ sec}. \tag{2}$$

For experts, we set $RT = 0$ sec.

For (d), we use a 27×27 matrix of digram frequencies to establish probabilities for each digram in common English, P_{ij}. The table includes the 26 letters plus the Space character. These are used to weight the movement time predictions in obtaining the mean movement time over all possible digrams:

$$\overline{MT} = \sum_i \sum_j P_{ij} \times \left(M_{ij} + RT \right) \tag{3}$$

RT is set to either .951 seconds (novices) or 0 seconds (experts), as noted earlier.[5]

Entry speed in wpm is calculated by taking the reciprocal of the mean movement time, multiplying by 60 seconds per minute, and dividing by 5 characters per word:

$$Entry\_Speed = \left(\frac{1}{\overline{MT}} \right) \times \frac{60}{5} \tag{4}$$

5. A recent experiment has revealed several weaknesses in the novice component of the model (MacKenzie & Zhang, 2000). Work is underway to refine the motor component of novice model to generate more accurate predictions.

The model takes particular care to accommodate the Space bar because it is the most prevalent character in text entry tasks. The result is a general behavioral description and predictive model of the task of text entry with a stylus and soft keyboard. We consider the predictions approximate but useful (for more details, see Soukoreff & MacKenzie, 1995).

This model has been subsequently used by others seeking an optimal keyboard layout for stylus typing (Hunter et al., 2000; MacKenzie et al., 1999; Zhai et al., 2000; Zhang, 1998). Their efforts are reported later in this article.

3.2. Language Prediction

Predictive text input techniques strive to reduce the input burden by predicting what the user is entering. This is accomplished by analyzing a large collection of documents—a corpus—to establish the relative frequency of characters, digrams (pairs of characters), trigrams, words, or phrases in the language of interest. These statistical properties are used to suggest or predict letters or words as text is entered. The seminal publication in the area of text prediction is by Shannon (1951), and although there are many ways to implement text prediction, most are based on this article.

Predictive input technologies have the capacity to significantly reduce the effort required to enter text—if the prediction is good. However, there are a few caveats to consider in basing a language model on a standard corpus, including (a) the corpus may not be representative of the user language, (b) the corpus does not reflect the editing process, and (c) the corpus does not reflect input modalities. An explanation of these points follows.

Corpus Not Representative of the User Language

The idea that a corpus is "representative of a language" is questionable when the domain is users interacting with computing technology. Users typically use a much richer set of characters and words than appear in any corpus, and the statistical properties in the user's set may differ from those in the corpus. A simple example is the Space key, which is the most common character in English text (Soukoreff & MacKenzie, 1995). However, the Space character is typically missing in tables of letter or digram probabilities used to build language models (e.g., Maynzer & Tresselt, 1965; Underwood & Schulz, 1960).

In addition, punctuation symbols are rarely included in letter or digram tables. Both Isokoski (1999) and Zhai et al. (2000) observed that some punctuation symbols occur more frequently than some of the less frequent letters. Inclusion of the Space character and simple punctuation symbols is the first step. We feel it is important to fully open the character set. (Corpora often do

not distinguish between capital and lowercase letters, but this is the special case of input modalities, discussed next.)

The characteristics of the text users enter are dependent on the application used to create the text. For example, we expect more formal prose using a word processor than an e-mail application. In addition, the type of application depends on the input device available—few people have the patience to enter volumes of text into a handheld PDA device. The kinds of text most likely entered in this context are short notes, phone numbers, URLs, acronyms, slang, and so forth, the statistical properties of which differ from formal English texts. Highly cryptic messages are common for text entry on cell phones (Grinter & Eldridge, in press).

Corpus Ignores the Editing Process

A corpus contains no information about the editing process, and we feel this is an unfortunate omission. Users are fallible, and the creation of a text message—or interaction with a system on a larger scale—involves much more than the perfect linear input of alphanumeric symbols. The input process is really the editing process.

Recently, we conducted a study to monitor and analyze keystroke-level interaction with desktop systems. Over a period of 2 months we logged all keystrokes (>400,000) for four desktop computer users. Figure 4 shows the 15 most common keystrokes. Common editing keys, such as Down, Back, and Up, figure very prominently in the table. Although mobile users engage a much different interface, the data in Figure 4 serve as a warning flag that input with computing technology, in general, is much richer than represented in a corpus.

Corpus Does Not Capture Input Modalities

Text documents do not reflect how they were created. For example, a corpus includes both capital and lowercase characters. In simple language models this distinction is ignored (e.g., *A* and *a* are considered the same). A more expansive model can easily accommodate this distinction simply by treating capital and lowercase characters as distinct symbols. However, from the input perspective, both approaches are wrong. Uppercase and lowercase characters are never entered via separate keys on a keyboard; thus, the seemingly more accurate treatment of uppercase and lowercase characters as distinct symbols is just as wrong.

For the user's interaction with the Shift and Caps Lock keys to be accommodated in a model of text input, activity with these and related keys should be included in the language model. In other words, it is the "language of inter-

Figure 4. **Relative frequency (%) of the 15 most frequent keystrokes from four users.**

S1		S2		S3		S4		All	
9.18	Space	10.42	Down	12.87	Space	8.75	Space	11.29	Space
7.14	Back	7.95	Space	8.69	Back	8.72	Back	7.10	Back
5.29	Down	5.57	Up	7.36	E	4.85	E	6.29	E
4.93	E	5.35	Shift	6.07	T	4.39	Down	5.11	T
4.19	A	5.33	Right	5.05	O	4.29	Return	4.29	O
3.85	Shift	4.49	Control	4.64	I	4.01	T	4.03	I
3.84	I	4.00	E	4.45	A	3.89	Shift	3.95	A
3.42	O	3.96	Left	4.18	S	3.88	O	3.94	Shift
3.28	T	3.73	Delete	4.16	N	3.83	I	3.78	S
3.27	R	3.25	T	3.79	R	3.57	R	3.57	N
3.22	N	3.12	S	3.46	Shift	3.31	A	3.33	R
2.98	Up	2.54	O	2.68	H	3.21	S	3.27	Down
2.92	Right	2.54	A	2.32	L	3.18	N	2.39	Delete
2.72	S	2.42	Back	2.24	C	2.84	D	2.32	H
2.48	Delete	2.38	I	2.14	D	2.26	H	2.22	C

action" that should be modeled. Note in Figure 4 that the Shift key fares no worse than 11th in the list of most-frequent keys.

3.3. Hybrid Input Techniques

Some text input techniques include both movement-minimizing and predictive features. Dasher (Ward et al., 2000) is a predictive text input technique using a pointing device to select from anticipated options (see also Ward, Blackwell, & MacKay, 2002). The options are presented to the user in boxes sized according to their relative probabilities. The boxes scroll and expand as the pointing device hovers near them (using graphics somewhat like a video game), allowing fast text entry. Thus, the technique is both movement minimizing and predictive. An online demonstration is available (http://wol.ra.phy.cam.ac.uk/mackay/dasher).

3.4. Key Minimization Techniques (Modes)

Because space is limited on small devices, keyboards that minimize the number of keys are of interest. However, users desire a large set of characters including the alphabet, numbers, symbols, and editing keys. An example of this is the standard PC-compatible 101-key keyboard. Although the standard PC keyboard has 101 keys, a user can produce closer to 800 individual keystrokes (each key is pressed in combination with Shift, Ctrl, or Alt, and the

Num Lock key changes the mode of the numeric keypad). The keys on the standard PC keyboard are, therefore, ambiguous; disambiguation is accomplished with the various mode keys.

There is another way to disambiguate keystrokes. Some keyboards are designed with more than one letter on each key (e.g., the alphabetic characters on a standard telephone keypad). Text entered on these is inherently ambiguous because different character strings correspond to the same key presses. For example, on a standard telephone keypad, both *gap* and *has* correspond to the key sequence 4–2–7. Disambiguation technology takes key press sequences and uses an embedded database of language statistics to identify legal words. These are presented to the user for verification. Automated disambiguation holds promise to increase the speed and accuracy of text input on ambiguous keyboards.

Conceptually, we can think of key ambiguity as a continuum (see Figure 5). At one extreme, we have a keyboard with a dedicated key for each symbol in the language (Figure 5a), whereas at the other extreme we have just one key that maps to every symbol in the language (Figure 5d). The keyboard in Figure 5d would be very fast[6] because only one key is pressed. However, it is of no practical use because each key press is ambiguous to the entire set of symbols in the language. Clearly, Figure 5d is little more than a curiosity. The Qwerty keyboard (Figure 5b) and telephone keypad (Figure 5c) represent two relevant points in the continuum.

The previous sections introduced many issues facing researchers in mobile text input, and we have delineated the design space within which this research takes place. In the following section we present a survey of mobile text entry techniques as found in research papers and commercial products.

4. SURVEY OF TEXT ENTRY TECHNIQUES

The survey is divided into key-based and stylus-based text input methods.

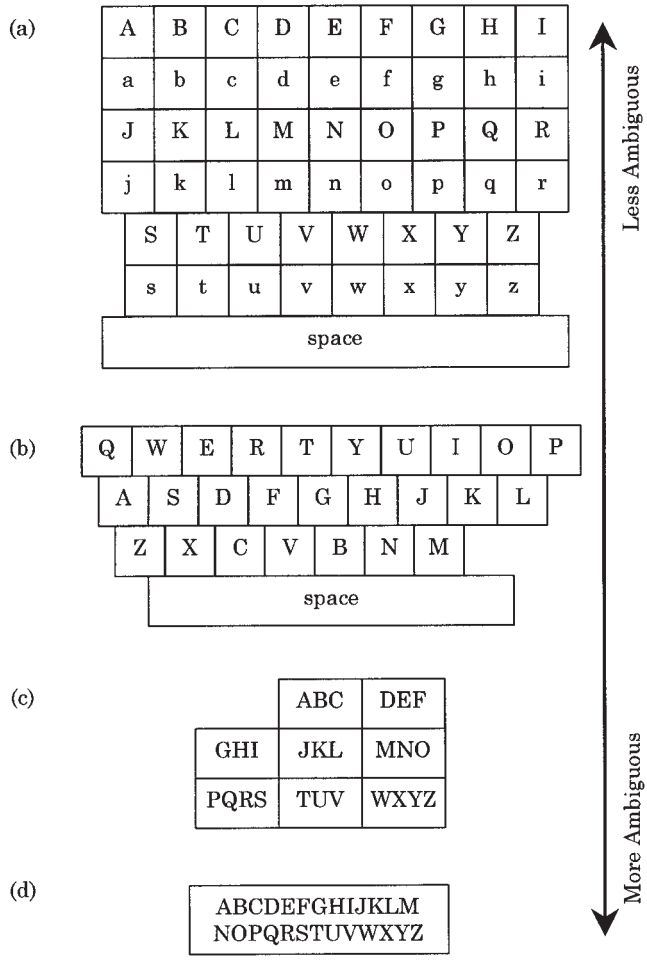
4.1. Key-Based Text Entry

Key-based text entry techniques range from those that use a keyboard where each key represents one or more letters to those with as few as three keys.

---

6. In fact, the text entry rate for this keyboard would be about 78.4 wpm. This figure is derived from the single finger key repeat time of .153 sec reported by Soukoreff and MacKenzie (1995). The text entry rate is (1 / .153) (60 / 5) = 78.4 wpm. The key repeat time may be as low as .127 sec (Zhai, Hunter, & Smith, 2000), and in this case, the upper bound is (1 / .127) (60 / 5) = 94.5 wpm.

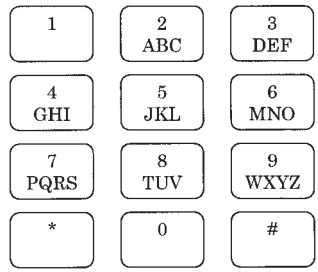
Figure 5. **The key-ambiguity continuum: (a) fictitious alphabetic keyboard with distinct keys for capital and lowercase letters; (b) Qwerty keyboard; (c) standard telephone keypad; (d) hypothetical single-key keyboard, which, to be useful, would require either many mode keys or a near psychic disambiguation algorithm.**



Telephone Keypad

The desire for an effective text entry method using the telephone keypad is fueled by the increase in text-messaging services and the movement toward consolidation of technologies such as wireless telephony and handheld computers. Text entry on a mobile phone is based on the standard 12-key telephone keypad (see Figure 6).

Figure 6. **The standard 12-key telephone keypad.**



The 12-key keypad consists of number keys 0 to 9 and two additional keys (* and #). Characters A to Z are spread over keys 2 to 9 in alphabetic order. The placement of characters is similar in most mobile phones, as it is based on an international standard (Grover, King, & Kuschler, 1998). The placement of the Space character varies among phones; however, it is usually entered with a single press of the 0 key or the # key. Because there are fewer keys than the 26 needed for the characters A to Z, three or four characters are grouped on each key, and so, ambiguity arises, as noted earlier. In the following paragraphs, we present three approaches to text entry on a phone keypad: multitap, two-key, and one-key with disambiguation.

The multitap method is currently the most common text input method for mobile phones. With this approach, the user presses each key one or more times to specify the input character. For example, the 2 key is pressed once for the character A, twice for B, and three times for C. Multitap suffers from the problem of segmentation, when a character is on the same key as the previous character (e.g., the word *on* because both O and N are on the 6 key). To enter the word *on*, the user presses the 6 key three times, waits for the system to timeout, and then presses the 6 key twice more to enter the N. Another segmentation technique is to use a special key to skip the timeout ("timeout kill"), thus allowing direct entry of the next character on the same key. Some phone models use a combination of the two solutions. For example, Nokia phones (Nokia Group, Finland; http://www.nokia.com) include both a 1.5-sec timeout and the provision for a timeout kill using the Down Arrow key. The user decides which strategy to use.

In the two-key method, the user presses two keys successively to specify a character. The first key selects the group of characters (e.g., the 5 key for J, K, or L). The second key specifies the position within the group. For example, to enter the character K the user presses 5 followed by 2 (K is second character in JKL). Although the two-key method is quite simple, it is not in common use for entering Roman letters. However, in Japan a similar method (often called the "pager" input method) is very common for entering Katakana characters.

A third way to overcome the problem of ambiguity is to add linguistic knowledge to the system. We call this technique *one-key with disambiguation*. An example is T9® by Tegic Communications, Inc. (Seattle, WA; http://www.tegic.com). When using T9 each key is pressed only once. For example, to enter *the*, the user enters 8–4–3–0. 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, linguistic disambiguation is not perfect because multiple words may have the same key sequence. In these cases the most common word 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 required 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 additional keys to obtain the desired word. Evidently, the term *one-key* in "one-key with disambiguation" is an oversimplification!

Silfverberg, MacKenzie, and Korhonen (2000) presented predictive models of these three text input methods based on the model of Soukoreff and MacKenzie (1995). They reported that the disambiguation of T9 works reasonably well, with expert predictions ranging from 41 to 46 wpm. However, these figures are coincident with rather broad assumptions. These include (a) all words entered are unambiguous, (b) users are experts (i.e., no typing, spelling, or other errors), and (c) all words entered are in the dictionary. Their predictions are, at best, an upper bound.

Many mobile phone manufacturers have licensed the T9 input technology, and since 1999 it has surfaced in commercial products (e.g., the Mitsubishi MA125, the Motorola i1000Plus, and the Nokia 7110). There is also a touch screen version of T9 that is available for PDAs. Bohan, Phipps, Chaparro, and Halcomb (1999) described an evaluation of the touch screen version.

T9 was the first disambiguating technology to work with a standard mobile phone keypad, but not the only such technology. Motorola's iTAP® is disambiguating technology similar to T9. Both iTAP and T9 support multiple languages. The Chinese version of iTAP uses a nine-key input method for writing the various strokes; it offers users more keystroke choices and is easy to learn (Sacher, 1998). Another similar technology is eZiText® by Zi Corporation

(Calgary, Alberta, Canada; http://www.zicorp.com). No published evaluations exist of iTAP or of eZiText.

A slightly different approach is presented in WordWise by Eatoni Ergonomics (New York; http://www.eatoni.com). To aid in disambiguation, a mode shift is used to explicitly choose one character from each key and the other characters remain ambiguous; this achieves partial disambiguation. Figure 7 illustrates the WordWise keypad.

The mode shift is implemented either with the 1 key (shown in Figure 7) or using a thumb-activated key on the side of the mobile phone. To enter the letter C, the Shift key is pressed followed by the 2 key. To enter the letter A, the 2 key is pressed by itself, and automatic disambiguation determines whether the user intended to enter A or B. The letters chosen for the mode shift are C, E, H, L, N, S, T, and Y, most of which are the most popular letters in each group (on each key). These letters were chosen to provide maximum separation for the disambiguation algorithm. One beneficial side effect of the mode shift is that words that are explicit (e.g., *the*, which is entered by holding Shift while entering 8–4–3) can be omitted in the internal database. This greatly reduces the memory requirements of the implementation—a critical factor for mobile phones.

Text input on the telephone keypad, working in concert with language-based disambiguation (e.g., T9 or WordWise), requires the attention of the user to monitor the outcome of keystrokes. A typical text creation task has two FOA because the user attends to both the keypad and the display. The performance impact of this behavior is difficult to model because it depends on cognitive and perceptual processes, and on user strategies (see Silfverberg et al., 2000, for further discussion).

With the multitap or two-key techniques, the outcome of keystrokes bears no such uncertainty; thus, skill in performing eyes-free input is more easily attained. The models created by Silfverberg et al. (2000) predict about 21 to 27 wpm for the multitap method and the two-key method.

Small Qwerty Keyboards

The most prevalent text input technology for low-end PDAs is the miniature Qwerty keyboard. There are many examples, such as the HP2000, some models of the HP Jornada, the Sharp (Osaka, Japan; http://sharp- world.com) Zaurus, the Sharp Mobilon, and the Psion (London; http://psion.com) Revo. Two-way pagers support text input and at least two companies have pager products with miniature Qwerty keyboards.

The BlackBerry by Research In Motion is a two-way pager with a small Qwerty keyboard (see Figure 8a). The keyboard is too small for touch typing, but it is suitable for one- or two-finger typing. Motorola has a similar product called the PageWriter (see Figure 8b).

Figure 7. **Eatoni Ergonomics WordWise keypad. The 1 key acts as a shift to explicitly select one letter on each alpha key.**

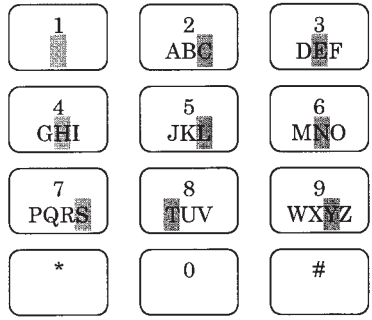
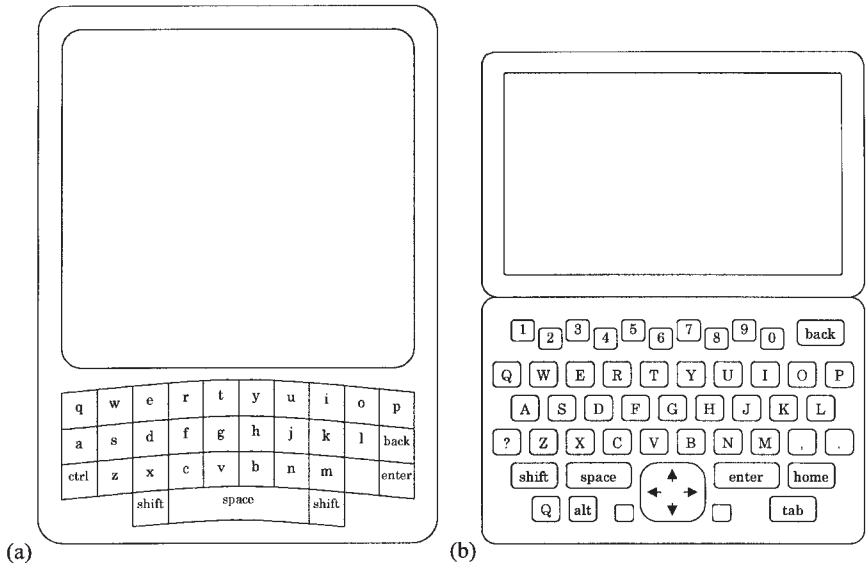


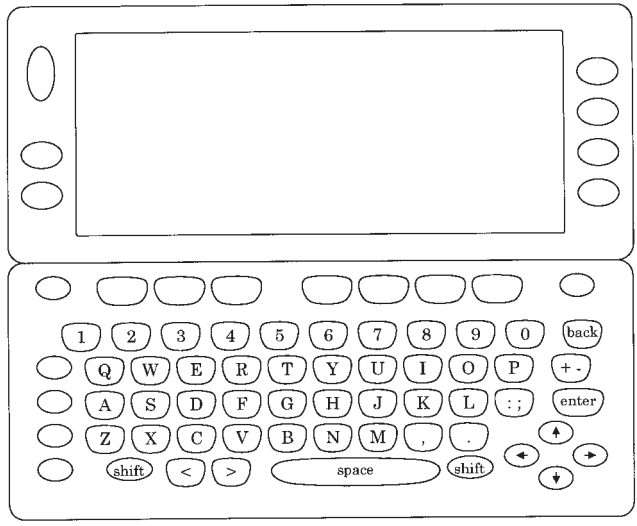
Figure 8. **(a) Research in Motion BlackBerry (RIM 957; actual size 79 × 117 mm) and (b) Motorola PageWriter 2000X (actual size 95 × 71 mm).**



The Nokia Communicator is a mobile phone with text-messaging functionality. It looks like a typical mobile phone when operated as a phone, but it opens to reveal a large LCD screen and miniature Qwerty keyboard inside (see Figure 9).

The BlackBerry, PageWriter, and Communicator are representative of small devices that have stayed with the Qwerty paradigm, and they are by no means alone. There are many similar devices on the market.

Figure 9. **Nokia 9110 Communicator (actual size 158 × 112 mm).**

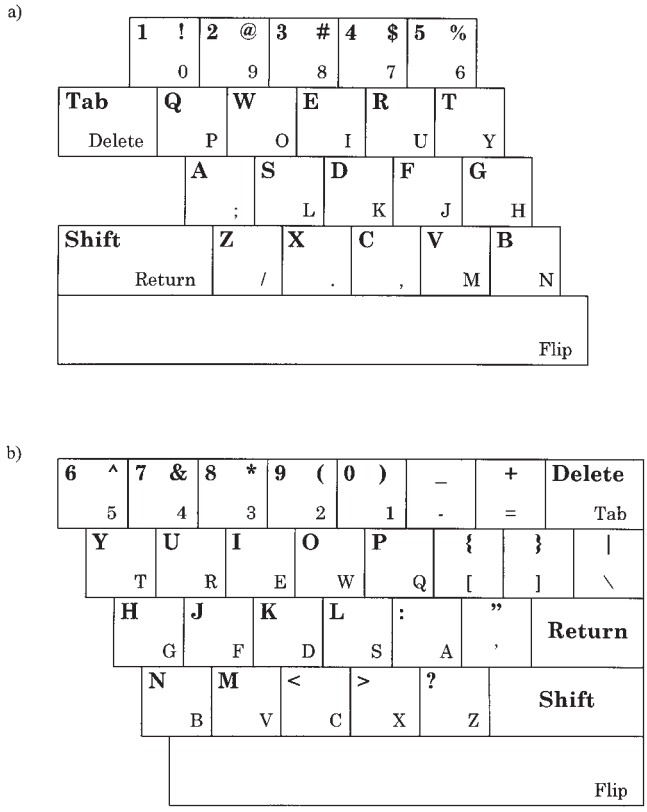


There is another way to reduce the size of a Qwerty-like keyboard. Matias and colleagues proposed a clever way to half the size of the keyboard and still leverage touch typing skills (Matias, MacKenzie, & Buxton, 1993, 1994, 1996a, 1996b). The Half-Qwerty keyboard, commercialized by Matias Corporation (Rexdale, Ontario, Canada; http://www.halfqwerty.com), is a regular Qwerty keyboard that is split in half. There are two possible Half-Qwerty keyboards—one corresponds to the left half of the Qwerty keyboard and the other to the right. To enter characters the user simply types the appropriate key in the regular fashion, but if the Space bar is held while a key is typed, the corresponding character from the other half of the keyboard is entered. Either hand can be used. Hitting the Space bar alone types a space. Note that the relative finger movements used for one-handed typing are the same as those used for two-handed typing. The two Half-Qwerty keyboards are depicted in Figure 10.

Matias and colleagues report the results of a rigorous user evaluation of the Half-Qwerty (Matias et al., 1993, 1996a). Right-handed participants using their left hands reached 50% of their two-handed typing speed after approximately 8 hr of practice, and after 10 hr all participants typed between 41% and 73% of their two-handed speed, ranging from 24 to 43 wpm.

The Half-Qwerty keyboard is unique among the other solutions to the mobile and handheld text entry problem because the keyboard is small, familiar to users, supports fairly rapid text entry, and has some significant applications. There are many industrial jobs that require a worker to enter text with one

Figure 10. **Matias Corporation Half-Qwerty keyboard. If implemented using a desktop keyboard, either half may be used.**



hand while doing another task with the other. The Half-Qwerty keyboard is also useful in situations where a user has lost the use of one hand. In both cases, software can be installed on a regular desktop computer that enables Half-Qwerty functionality. A small stand-alone version is now available as an add-on for handheld devices.

Although lugging around a full-sized Qwerty keyboard to use with the PDA in one's shirt pocket seems odd, collapsible Qwerty keyboards allow users to do just that. In 1999 Think Outside Inc. (Carlsbad, CA; http://www.think-outside.com) released the Stowaway™, a full-size Qwerty keyboard that collapses to a 91 × 130 × 20 mm volume. Originally released for the Palm, the Stowaway was later adopted by Palm Computing, becoming the Palm™ portable keyboard. Think Outside also produce collapsible keyboards for other families of PDA.

Five-Key Text Entry

By way of introduction to five-key text input, we mention the *date stamp* method (also known as the *three-key* method). This method can be implemented using very limited hardware: technology to display at least one character, two buttons (or a wheel) to scroll through the alphabet, and an Enter key. It is called the date stamp method because, similar to a date stamp, the desired character is selected by rotating through the character set. Video arcade games often use this technique for players to enter their name when they achieve a high score. The technique is also commonly used for entering text into some electronic musical instruments. Although the three-key method is reasonable for entering small amounts of text into devices with a simple interface, the method is frustratingly slow and not suitable for even modest amounts of text entry (see Bellman & MacKenzie, 1998, for further discussion).

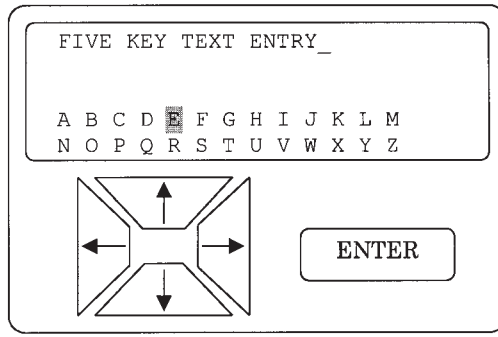
Five-key text entry uses an interface with four cursor keys (up, down, left, and right) and an Enter key (see Figure 11). The alphabet, number, and symbol characters are presented on a LCD display with typically three to five rows and 10 to 20 columns, and the five keys are used to move a cursor and select one letter at a time.

The characters are presented in alphabetic order or in the familiar Qwerty arrangement. The five-key input method is typically used on very small devices like the recent generation of pagers, which only have enough space for a small LCD screen and five keys. An example is the AccessLink® II pager from Glenayre Electronics Inc. (Charlotte, NC; http://www.glenayre.net).

The main problem with the five-key method is that many key presses are required to move between characters, and this significantly slows input. In view of this, Bellman and MacKenzie (1998) devised a technique known as *fluctuating optimal character layout* (FOCL). The idea is that because the input device knows the last character the user has entered, it can subsequently present the characters in an arrangement that places the most likely characters closer to the cursor's home position; the display is rearranged after each character entered so as to minimize the number of cursor movements to select the most likely next character. They show that the average number of KSPC can be reduced by over 50%, from just over 4 KSPC for the alpha layout to less than 2 KSPC using FOCL.

Bellman and MacKenzie (1998) reported the results of an exploratory study comparing FOCL to the five-key input method using the Qwerty arrangement of letters. Their study, with 10 participants, found that after 10 sessions of 15 min each there was no statistically significant difference in entry speed or accuracy. The average speed they reported for both Qwerty and FOCL is 10 wpm. Although the study was longitudinal in nature, evidently participants did not have enough exposure to FOCL to approach their maximum text entry

Figure 11. **Five-key text entry.**



speeds. Although fewer keystrokes were required to enter each character, more visual scan time was required to find the next character. In short, as with many other optimized text entry methods, the advantage of the input technique is apparently not realized until users invest considerable time to become familiar with the new technology.

The three-key, five-key, and FOCL text input techniques all require the user's attention on the screen to scroll around and select characters. Therefore, text creation is a two-FOA task with these techniques. Single-handed input is possible with all of these input techniques.

Other Small Keyboards

Some researchers have proposed alternatives to the telephone or Qwerty key arrangements. The *single-hand key card* (SHK) is a small card with a keyboard and joystick proposed by Sugimoto and Takahashi (1996). The SHK is held in one hand, pinned between the palm and the thumb in such a way that the four fingers manipulate the keyboard and joystick on the top face of the device. SHK is a small keyboard with multiple characters on each key. It employs disambiguation technology. The keyboard arrangement of SHK appears in Figure 12. The joystick and three function keys appear in a row above the keyboard on the device (not shown). The AR key in Figure 12 toggles through the word possibilities generated by the ambiguity resolution feature.

Sugimoto and Takahashi (1996) reported that the keys were arranged so as to reduce the average motion of the fingers, although they have not explained in detail how they came to their key arrangement or published an evaluation of their device. Once the arrangement is learned by the user, the device could support single FOA text creation and single-handed text input.

Another important class of keyboard is *chording keyboards*, where text is entered by pressing multiple keys simultaneously. Because multiple keys are

Figure 12. **The key arrangement of the Single-Hand Key card device.**

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

pressed, fewer keys are needed on a chord keyboard (resulting in smaller devices) and chords not being used for entering single letters can be used to enter words. The Twiddler by Handykey Corporation (Denver, CO; http://www.handykey.com) is a chord keyboard popular with researchers in the wearable and ubiquitous computing fields. The Twiddler is operated with one hand and has 4 mode keys depressed by the thumb (Number, Alt, Ctrl, and Shift) and 12 keys for the fingers. The Twiddler keyboard appears in Figure 13; notice that the layout of the characters is somewhat alphabetical. However, the Twiddler is user configurable; the user may change the characters (or words) entered by each chord (the keyboard appearing in Figure 13 is the default layout), and other character mappings for the chords have been proposed for the Twiddler, which are claimed to map common characters to easier chords. The Twiddler also has chords defined for common small English words and parts of words (e.g., *the*, *and*, *-ion*, and *-ing*, etc.).

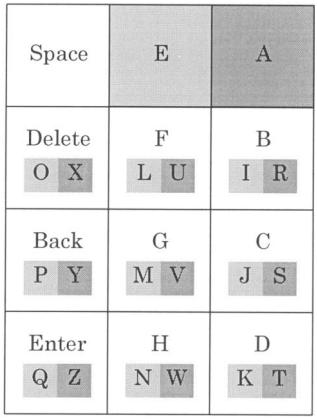
The Twiddler is by no means the only chord keyboard. Other examples include the BAT™ by Infogrip, Inc. (Ventura, CA; http://www.infogrip.com) and MonoManus® by ElmEntry Enterprises (Minneapolis, MN; http://www.hankes.com/eee/index.htm). However, most of these keyboards interface to desktop computers and are not expressly for mobile computing platforms. The Twiddler interfaces to the Palm only if the Happy Hacking™ Cradle (by PFU America, Inc; San Jose, CA; http://www.pfuca.com) is used (available separately).

The Twiddler can be used with one hand (zero FOA, once the chords are learned), and anecdotal reports of typing speed as fast as 50 wpm have been reported (Hjelm, Tan, Fabry, Fanchon, & Reichert, 1996).

4.2. Stylus-Based Text Entry

Stylus-based text entry uses a pointing device, typically a pen (a.k.a. stylus), to select characters through tapping or gesture. Although our discussions here are limited to stylus input, there are several related examples of research in

Figure 13. **The key arrangement of the Twiddler chord keyboard device. Letters with a white background are entered by pressing the key by itself. Letters with a light grey background are entered by pressing the key and the E key simultaneously. Letters with a dark grey background are entered by pressing the key and the A key simultaneously.**



mobile text entry using finger or touch input, wherein the user's finger is used instead of a stylus (e.g., Enns & MacKenzie, 1998; Fukumoto & Suenaga, 1994; Goldstein, Book, Alsio, & Tessa, 1999). All of the stylus-based text entry techniques require two hands, unless the user can support the device on a table while using it.

Traditional Handwriting Recognition

Handwriting recognition was once touted as *the* solution for mobile text entry, but early systems received considerable bad press, as noted earlier. To be fair, handwriting recognition is a difficult problem, and the technology has improved since the early days. There are two problems that handwriting recognizers must solve: segmentation and recognition. The input to a recognizer is a series of ink trails, with each stored as a set of digitized points representing the stylus travel between pen-down and pen-up actions. Segmentation is the process of determining which segments are in which characters. With the goal of supporting "natural handwriting," input is often a mixture of block printing and cursive handwriting. As one might imagine, segmenting the strokes in the sloppy scrawl of a user is very difficult indeed. One way to reduce the complexity is to constrain input (e.g., to support block printed characters only). However, entry like this is by no means "natural." Generally, the more relaxed the constraints, the more difficult the segmentation and recognition process; recognition accuracy usually suffers. To compensate,

recognizers are made more complex and, unfortunately, require more memory (see Tappert, Suen, & Wakahara, 1990, for a detailed survey of recognition techniques and technologies).

One obstacle for recognition-based technologies is high user expectations. LaLomia (1994) reported that users are willing to accept a recognition error rate of only 3% (a 97% recognition rate), although Frankish and colleagues (1995) concluded that users will accept higher error rates depending on the text-editing task. Several researchers have published studies evaluating or comparing the recognition rate of various recognition systems. Chang and MacKenzie reported a recognition rate of 87% to 93% for two recognizers (Chang & MacKenzie, 1994; MacKenzie & Chang, 1999). Wolf, Glasser, and Fujisaki (1991) reported a recognition rate of 88% to 93%. Santos, Baltzer, Badre, Henneman, and Miller (1992) reported a novice recognition rate of 57%, although this improved to 97% after 3 hr of practice. These studies suggest that recognition technology is close to matching user expectations for expert users but that novices may be discouraged by their initial experiences. Perhaps the acid test, an observation suggesting that handwriting recognition does not yet perform adequately, is that there are no mobile consumer products in the market today where natural handwriting recognition is the sole text input method. The products that do support stylus-based text input work with constraints or stylized alphabets (see later).

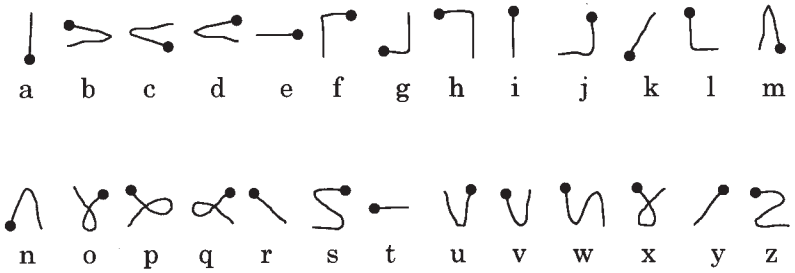
Gibbs (1993) made an important observation on text entry speed and handwriting recognition. In his summary of 13 recognizers, the recognition speed of the systems was at least 4 cps, which translates into 48 wpm. However, human hand-printing speed is typically on the order of 15 wpm (Card et al., 1983; Devoe, 1967; Van Cott & Kinkade, 1972). In other words, speed is a function of human limitations, not machine limitations. Even with perfect recognition, therefore, entry rates can never reach those of, for example, touch typing.

Unistrokes

Unistrokes is a stylized single-stroke alphabet developed by Goldberg and Richardson at the Xerox Palo Alto Research Center (Goldberg & Richardson, 1993). At the time of the invention, handwriting recognition technology was not in good stead with users, as the problems noted earlier were rampant in existing products. To address these Goldberg and Richardson developed a simplified set of strokes that is both easier for software to recognize and quicker for users to write. The Unistrokes alphabet appears in Figure 14.

The name *Unistrokes* describes the most significant simplification that Goldberg and Richardson (1993) made: Each letter is written with a single stroke. This greatly simplifies recognition, as the segmentation problem is essentially eliminated. The strokes are so simple that users can write Unistrokes

Figure 14. **Unistrokes alphabet.**



without watching the stylus. Goldberg and Richardson observed that Unistrokes afford what they termed *heads-up text entry* (i.e., reduced FOA). The Unistrokes alphabet does not contain numbers, punctuation, or symbolic characters, although the original publication (Goldberg & Richardson, 1993) suggests ways of supporting these (e.g., using a dedicated stroke as a mode shift).

Although a comparative study of Unistrokes has never been undertaken, some experimental results are given (Goldberg & Richardson, 1993). Ignoring errors, a text-entry rate of 2.8 cps (i.e., 34 wpm) was reported.

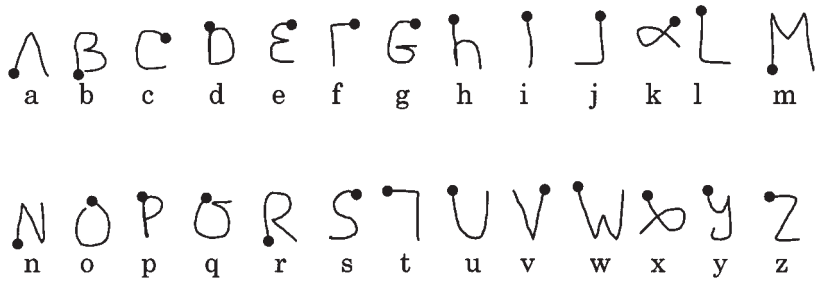
Although an interesting and promising idea, Unistrokes did not catch on, and the most likely reason is that the strokes are not similar enough to regular handwritten or printed letters; the strokes must be learned. Palm designed a single-stroke system called Graffiti that is used in their Palm product. Graffiti has been credited as a significant reason for the commercial success of the Palm (Blickenstorfer, 1995). The Graffiti alphabet appears in Figure 15.

Graffiti has strokes for punctuation, numbers, symbolic characters, and mode switches (capital vs. lowercase). These are omitted in Figure 15 for brevity. Capital and lowercase characters are supported with mode switching, which is accomplished with a dedicated stroke. Basic editing (Backspace) is also supported with a special stroke.

The great advantage that Graffiti has over Unistrokes is its similarity to normal hand-printed characters. MacKenzie and Zhang (1997) performed a study of the immediate usability of Graffiti. They observed that 79% of the Graffiti strokes match letters of the Roman alphabet. Under experimental conditions they measured the accuracy with which participants could enter the alphabet following 1 min of studying the Graffiti reference chart, following 5 min of practicing with Graffiti and following a 1-week lapse with no intervening practice. The accuracies they reported were very high–86%, 97%, and 97%, respectively.

Isokoski (1999) presented a single-stroke alphabet that can be entered using a wide range of pointing devices. He observed that the easiest motions to make with pointing devices are the four primary compass directions: up, down, left, and right. Another design objective was finding the optimal mapping between

Figure 15. **Graffiti alphabet.**



the four directional strokes and the characters of the alphabet (more frequent characters should have shorter strokes). He called the result *minimal device-independent text-input method* (MDTIM). The MDTIM alphabet appears in Figure 16.

Isokoski (1999) evaluated his single-stroke alphabet with a variety of pointing devices. The measured average text entry speed using a touch pad was 7.5 wpm. The study was not longitudinal, and the participants were still showing improvement at the end of the trials. The MDTIM alphabet suffers from the same affliction as Unistrokes and many other text input methods: The alphabet is not familiar to the average user, and practice is required to learn the alphabet and attain fast entry speeds. However, Isokoski's results do indicate that the MDTIM alphabet is indeed device-independent.

Until recently Windows CE devices were without a similar easy-to-learn handwriting recognition technology. This changed in 1998 when Microsoft licensed Jot from Communication Intelligence Corporation. Jot recognizes many of the Graffiti strokes and a number of alternative strokes similar to normal handwriting and printing as well. The Jot alphabet appears in Figure 17.

Jot also includes strokes for numbers, symbolic characters, and common editing functions. The different cases (capital vs. lowercase) are selected by where the user writes the stroke on the touch screen of the device. Jot also allows some customization: Users can indicate writing preferences for some characters.

All the alphabets just described have the potential to support single-FOA text entry once the user is familiar with the stylized alphabet.

Gesture-Based Text Input

Gestures are informal motions for communication. We classify the text entry methods in this section as gestural because of their informality and fluidity. Character-recognition-based and soft-keyboard-based input techniques have fixed characters that are entered in a certain way, or the stylus must be tapped in a certain location to select characters for input. Gesture-based text input

Figure 16. **MDTIM alphabet.**

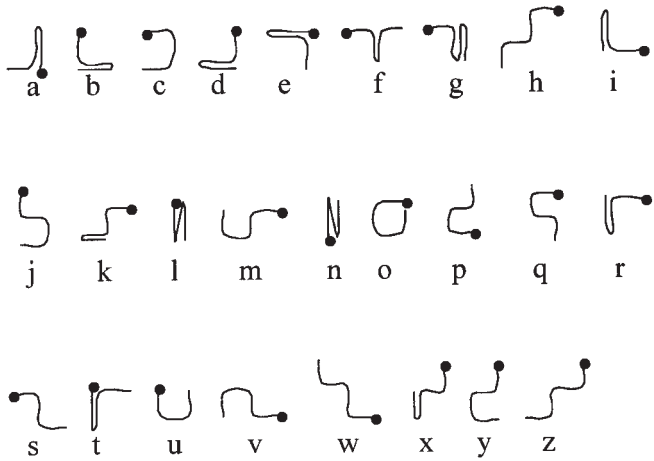
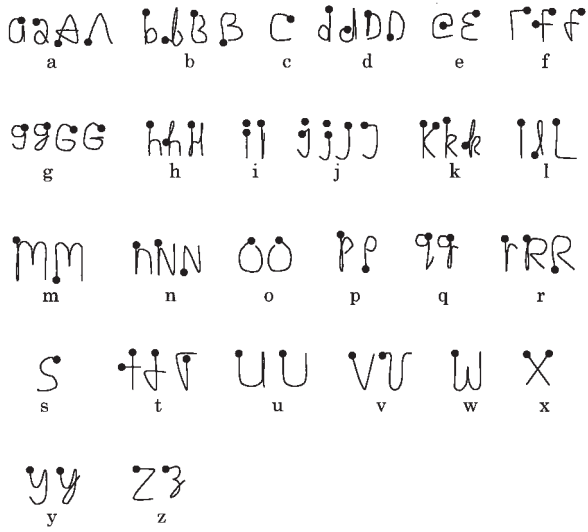


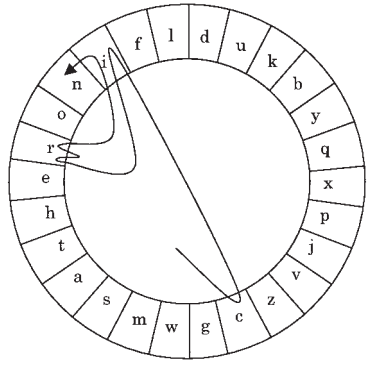
Figure 17. **Jot alphabet.**



technologies do not have a fixed set of strokes that a recognizer turns into characters; gestural text input methods have a framework in which informal stylus motions are interpreted as characters.

An example of this is Cirrin, a technology presented by Mankoff and Abowd (1998). The letters of the alphabet are arranged inside the perimeter of an annulus. Figure 18 shows the word *cirrin* written on the Cirrin interface.

Figure 18. **The Cirrin interface, indicating how to enter the word *cirrin*.**



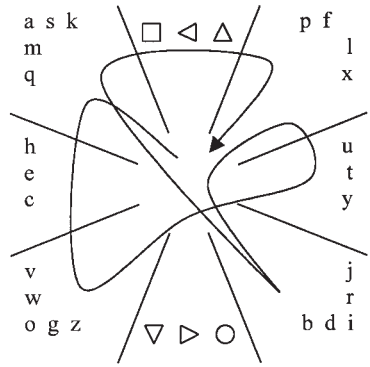
Characters are selected by moving into and out of the appropriate sector of the annulus. Mankoff and Abowd chose the circular arrangement and the order of the letters to minimize the distance between likely consecutive characters. In Figure 18, notice how the final two letters, *in*, are selected; the stylus can be moved directly from one letter into the neighboring letter.

Cirrin is not a "heads-up" text input method; users must attend to the interface when entering text. As presented, only alphabetic characters are supported. A space is entered by lifting the stylus and punctuation and mode shifts are accomplished by using an auxiliary technique, such as keys operated by the nondominant hand.

Mankoff and Abowd (1998) did not report a user evaluation in their publication; however, they stated that Cirrin "is about as fast as existing pen entry systems" (p. 214), but no indication is given of specifically what pen entry systems they compared Cirrin to.

Quikwriting is an input technology described by Perlin (1998). The idea is to have a 3×3 grid where characters are entered with strokes that begin in the center "home" position and move through one to three adjoining positions, returning back to the home position. Figure 19 illustrates the Quikwriting lowercase menu. Quikwriting has similar displays and modes for numbers, capitals, and symbols. The symbols in the top center and bottom center positions represent the different modes. Letters that occur more frequently in English are given the shortest strokes. For example, *i* in Figure 19 is selected by moving into the bottom right position and then returning back to the home position. Infrequent letters have longer strokes (*k* requires a move into the upper left position, then to the upper right position, and finally back to home). There is an online demonstration available at http://www.mrl.nyu.edu/projects/quikwriting/. Quikwriting, like Cirrin, requires the user to look at the interface and so is a two-FOA interface, if users correct errors as they go.

Figure 19. **Quikwriting lowercase interface indicating how to enter the word *quik*.**



At the time of Perlin's (1998) publication, a user evaluation had not been performed, although he wrote that users familiar with Graffiti found Quikwriting about three times faster.

Another gestural text input technology is T-Cube, described by Venolia and Neiberg (1994). T-Cube is similar to a two-tier pie menu system. Figure 20 shows the pie menu structure of T-Cube. The user places the stylus within one of nine starting positions (arranged in a 3×3 grid; see Figure 20a). The location where the stylus is first placed indicates which of the pie menus (see Figure 20b) the user will select a character from. One of the eight characters in the submenu is chosen by flicking the stylus in the appropriate direction. The interface does not display the pie menus (Figure 20b) unless the user hesitates. T-Cube includes numbers, many symbol characters, and basic backspace editing. Like the other gestural input techniques, T-Cube requires the attention of the user, making standard text entry a two-FOA task.

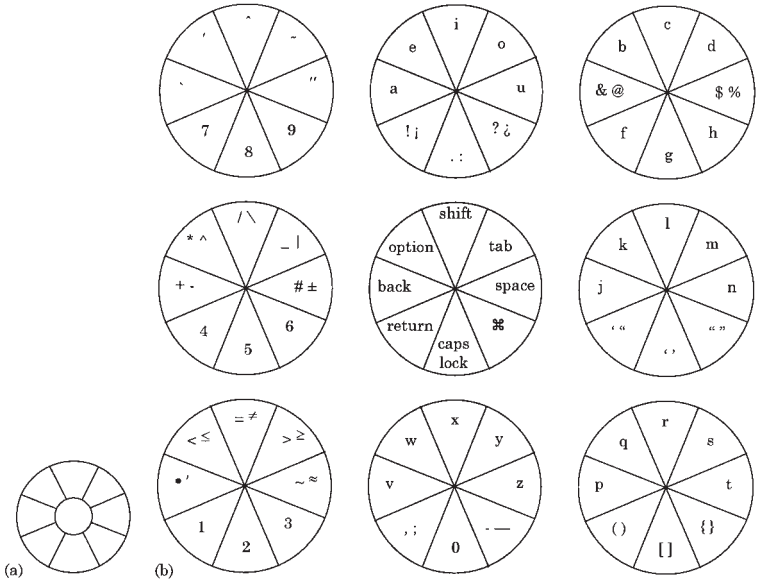
Venolia (1994) presented the results of a user study of T-Cube indicating that reasonably fast text entry can be achieved; one of his participants achieved a rate of 106 characters per minute (21 wpm). However, he also acknowledged that the interface is difficult to learn.

Soft Keyboards

A soft keyboard is a keyboard implemented on a display with built-in digitizing technology. Text entry is performed by tapping on keys with a stylus or finger. However, eyes-free entry is not possible. The advantages of soft keyboards include simplicity and efficient use of space. When no text entry is occurring, the soft keyboard disappears, thus freeing screen space for other purposes.

Soft keyboards have performance advantages too. MacKenzie, Nonnecke, Riddersma, et al. (1994) reported a text entry task comparing a Qwerty soft

Figure 20. **T-Cube pie menu structure: (a) first-level menu; (b) second-level menus.**



keyboard, an ABC soft keyboard, and hand printing. The Qwerty soft keyboard was both faster and more accurate than hand printing (see Figure 3).

This section presents some variations of soft keyboards developed in industry and research labs. We begin by giving the predicted expert entry rates according to the model of Soukoreff and MacKenzie (1995) presented earlier. These are given in Figure 21 sorted by predicted entry rate, highest to lowest. Several changes and one correction[7] have been introduced to the model since

7. The model works by using digrams to model the users' transitions from key to key as they enter text. However, a long Space key (such as in the Qwerty keyboard) or multiple Space keys are best modeled with trigrams. The error made by MacKenzie and Zhang (MacKenzie & Zhang, 1999; Zhang, 1998) was a miscalculation involving the relative probabilities of trigrams containing a Space character. Typically, trigram frequencies are not explicitly represented but, rather, are derived from digram frequencies. The probability of a trigram (i.e., the probability of the character sequence *i–j–k*) is found with the expression:

$$P(i,j,k) = P(i,j)\frac{P(j,k)}{\sum_{s} P(j,s)},$$

where *P(i,j)* is the probability of digram *i–j*. MacKenzie and Zhang omitted the denominator from their calculations. This error was first reported by Hunter, Zhai, and Smith (2000) and Zhai, Hunter, and Smith (2000).

Figure 21. **Expert predictions for various soft keyboard layouts.**

Keyboard Layout	Expert Prediction (wpm)	Improvement (%) Over Qwerty	Figure
Metropolis II	42.94	42.9	Figure 26c
OPTI II	42.37	41.0	Figure 25b
OPTI I	42.16	40.3	Figure 25a
Metropolis I	42.15	40.3	Figure 26b
Fitaly	41.96	39.7	Figure 23
Hook's	41.15	37.0	Figure 26a
Cubon	37.02	23.2	Figure 24
Lewis	34.65	15.3	Figure 27
ABC III	32.50	8.2	Figure 22c
ABC IV	30.18	0.5	Figure 22d
ABC II	30.13	0.3	Figure 22b
Qwerty	30.04	—	Figure 5b
DotNote	29.46	–1.9	Figure 28
ABC I	28.79	–4.2	Figure 22a

Note. wpm = words per minute.

it was introduced in 1995. The entries in the table are updated from earlier publications to reflect these changes (see also Zhai et al., 2000, and elsewhere in this issue of *Human–Computer Interaction*) for a discussion of the model's sensitivity to factors such as the coefficients in the Fitts' law model and the corpus used in building the language model.

Figure 21 also gives the improvement of each soft keyboard, relative to Qwerty. At the top of the list is the Metropolis II keyboard, with a predicted text entry rate 42.9% higher than Qwerty. We visit this shortly.

There are two keyboard arrangements generally familiar to most users: Qwerty and alphabetic. The Qwerty keyboard was shown earlier (see in Figure 5b). A few alphabetic arrangements appear in Figure 22. An experiment reported by MacKenzie et al. (1999) found that participants achieved rates of 20 wpm on a Qwerty soft keyboard and 11 wpm using an ABC layout (ABC I in Figure 22a). The predicted expert entry rates are 30.04 wpm for a Qwerty soft keyboard and 28.79 wpm for the ABC I arrangement. Predictions for the ABC II, ABC III, and ABC IV arrangements are 30.13 wpm, 32.50 wpm, and 30.18 wpm, respectively (see Figure 21).

The inventors of the FITALY keyboard by Textware™ Solutions Inc. (Burlington, MA; http://www.textwaresolutions.com) used an ad hoc optimization approach to minimize the distance between common character pairs. The resulting keyboard (see Figure 23) contains two Space bars and the letters are arranged so that common pairs of letters are often on neighboring keys. MacKenzie et al. (1999) reported a walk-up (i.e., participants did not have previous experience and did not get much practice) typing rate for the

Figure 22. **Some alphabetic keyboard arrangements (a) ABC I; (b) ABC II; (c) ABC III; (d) ABC IV.**

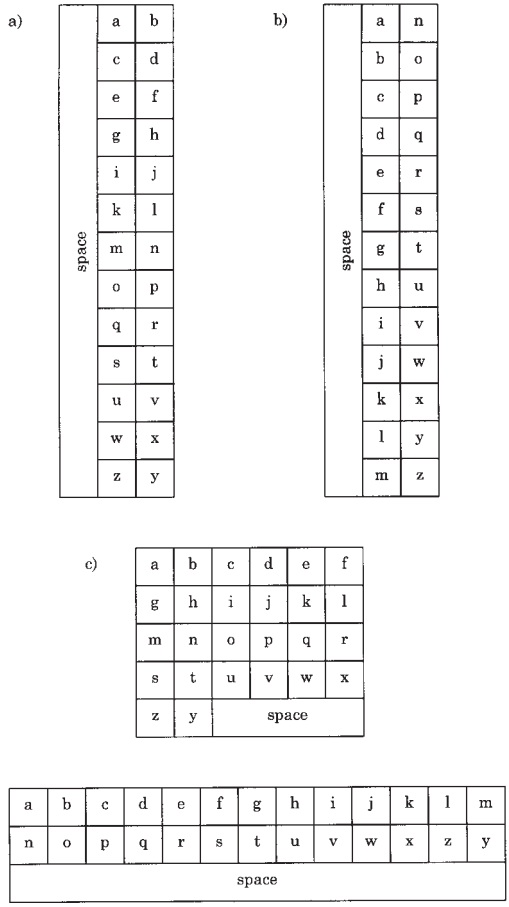


Figure 23. **FITALY keyboard.**

z	v	c	h	w	k
f	i	t	a	l	y
space		n	e	space	
g	d	o	r	s	b
q	j	u	m	p	x

FITALY keyboard of 8 wpm. The expert prediction for the FITALY layout is 41.96 wpm (see Figure 21).

Little information is available on the Cubon keyboard, except that it seems to have been proposed by R. A. Cubon and is used in rehabilitation situations for persons with the use of only one finger, or with a head-mounted pointing device. We know of no published user studies. The Cubon keyboard arrangement that appears in Figure 24 is given in Zhai et al. (2000). The expert prediction for Cubon is 37.02 wpm.

MacKenzie and Zhang (1999) used Soukoreff and MacKenzie's model to produce an optimized keyboard arrangement. The OPTI I keyboard appears in Figure 25a. They reported a predicted expert typing rate of 58 wpm; however, this prediction includes the error pointed out by Zhai and colleagues (see Footnote 7). Our current prediction for the OPTI I layout stands at 42.16 wpm.

MacKenzie and Zhang performed a longitudinal study over 20 sessions comparing the Qwerty and OPTI I arrangements and found that the average typing rate for OPTI I increased from 17 wpm initially to 44 wpm after 8 hr of practice (see Figure 1). For the Qwerty layout, rates increased from 28 wpm to 40 wpm over the same interval. The average rates for OPTI I exceeded those for the Qwerty layout after about 4 hr of practice.

The alert reader will notice that something is amiss: The *observed* rates actually exceeded the expert predictions! The most likely explanation is that the slope coefficient in the Fitts' law prediction model is too conservative. The slope coefficient used in the predictions is .204 sec per bit (see Equation 1), a value obtained from a pointing device study using a stylus on a Wacom tablet in a serial tapping task (MacKenzie, Sellen, & Buxton, 1991). The reciprocal of the slope coefficient is commonly known as the Fitts' law bandwidth and, in this case, is $1 / .204 = 4.9$ bits per second. A discrepancy of even 1 bit per second is enough to raise the predicted rate above the observed rates.[8] Although the model is clearly sensitive to the slope coefficient in Fitts' law, adjustments do not change the rank order of the predictions in Figure 21. The reader is directed to Zhai et al. (2000) and elsewhere in this issue of *Human–Computer Interaction* for further discussion.

---

8. The bandwidth coefficient in MacKenzie, Sellen, and Buxton (1991) was measured in an "indirect" task: Participants manipulated the stylus on a Wacom tablet while attending to the system's display. Stylus tapping on a soft keyboard is a "direct" task: Participants manipulate the stylus on the soft keyboard while also visually attending to the soft keyboard. This, alone, is cause to suspect that the bandwidth coefficient used in Soukoreff and MacKenzie's model is conservative. Although no experiment measuring the bandwidth coefficient for stylus tapping on a soft keyboard has been published, we suspect such an experiment would yield a higher bandwidth. The effect would be to increase all the predictions in Figure 21.

Figure 24. **Cubon keyboard (from Zhai, Hunter, & Smith, 2000).**

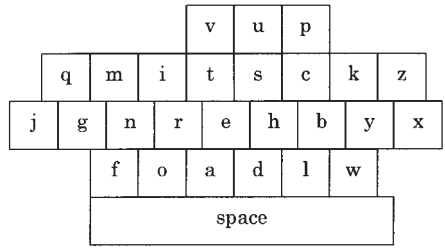


Figure 25. **(a) OPTI I; (b) OPTI II soft keyboards. From "The design and evaluation of a high-performance soft keyboard," by I. S. MacKenzie and S. X. Zhang, 1999, *Proceedings of the CHI 99 Conference on Human Factors in Computing Systems.* Copyright 1999 by ACM. Reprinted with permission. From *A high performance soft keyboard for mobile systems*, by S. X. Zhang, 1998, Unpublished thesis, University of Guelph, Ontario, Canada. Copyright 1998 by S. X. Zhang. Reprinted with permission.**

a)

q	f	u	m	c	k	z
space		o	t	h	space	
b	s	r	e	a	w	x
space		i	n	d	space	
j	p	v	g	l	y	

b)

q	k	c	g	v	j
space	s	i	n	d	space
w	t	h	e	a	m
space	u	o	r	l	space
z	b	f	y	p	x

In follow-up work, Zhang (1998) proposed a slight modification to OPTI I. The OPTI II appears in Figure 25b and yields an expert prediction of 42.37 wpm (see Figure 21).

Hunter et al. (2000) and Zhai et al. (2000) applied two physics-inspired techniques to the model of Soukoreff and MacKenzie and generated optimal keyboards. They used a mechanical simulation of a mesh of springs, where the springs were stretched between the characters of the alphabet and tensioned proportionally to digram probabilities in English. The technique is an application of a greedy algorithm to reduce the physical distance between more likely character pairs.[9] The result is a keyboard they call Hook's key-

9. The term *greedy algorithm* refers to a class of algorithms for solving minimization (or maximization) problems. Greedy algorithms try to find the minimum solution of a problem by always moving in the direction of steepest descent. However, greedy algorithms can become trapped in "local minimums." This is analogous to searching for the deepest point in a valley by always walking downhill—but becoming trapped in a hole in the side of the valley because a step upward (to a higher altitude) would be required to leave the hole.

board after Hook's law[10] (see Figure 26a). It yields a predicted expert entry rate of 41.15 wpm.

A second approach they took was to apply the Metropolis algorithm, which is theoretically more appealing because it employs a random-walk strategy instead of a greedy algorithm.[11] The Metropolis I and Metropolis II keyboards have higher predicted expert typing speeds and appear in Figure 26b and Figure 26c. In their first publication reporting preliminary results (Hunter et al., 2000), they presented a Metropolis keyboard, which we denote Metropolis I. In a later publication (Zhai et al., 2000), another Metropolis-derived keyboard was presented, which we call Metropolis II. The predicted expert entry rates for the Metropolis I and Metropolis II keyboards are 42.15 wpm and 42.94 wpm, respectively. Metropolis II has the distinction of yielding the fastest predictions of any soft keyboard tested (see Figure 21). Longitudinal evaluations of the Hook's, Metropolis I, or Metropolis II keyboards have not been undertaken, so the entry speeds attainable in practice are not known.

Lewis et al. (1999a, 1999b) also tried to optimize entry rates for a soft keyboard. They applied network analysis to character pair probabilities to determine the most strongly associated pairs. Then, using an ad hoc method to minimize distances for the strongly associated character pairs, they produced the keyboard arrangement in Figure 27. Lewis et al. performed a comparative user evaluation but they did not report their results; estimating from their published report (Lewis et al., 1999b, Figure 1) suggests they measured typing speeds of 25 wpm for the Qwerty control condition and 13 wpm for their keyboard design. They also reported that when asked, participants indicated a preference for the Qwerty layout. Our expert prediction for the Lewis keyboard is 34.65 wpm (see Figure 21).

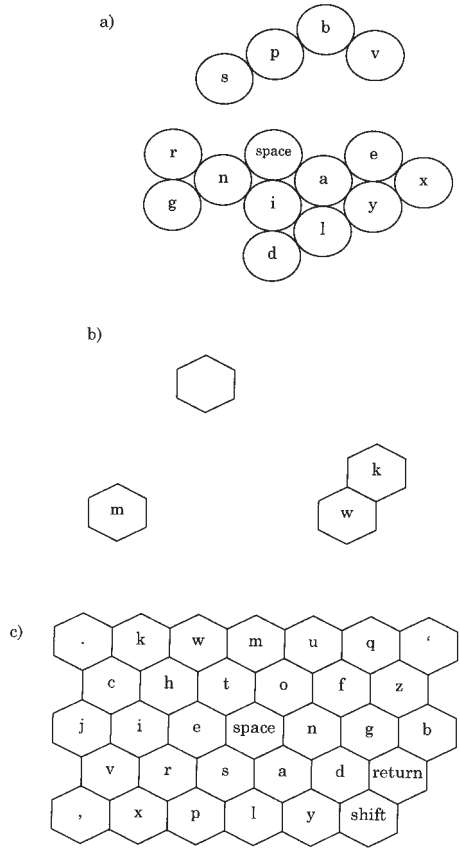
The DotNote keyboard by Útilware (http://www.utilware.com) was designed to support single-handed text entry on the Palm as an alternative to the built-in Graffiti handwriting recognition, which requires two hands (one to hold the device, the other to manipulate the stylus). To support finger or

---

10. Hook's law states that the tensional force in a spring is proportional to its extension, that is, how much it has been stretched from its equilibrium length, or, $F = -kx$, where F is the force of the spring, x is the distance that the spring has been stretched, and k is the spring constant, which varies from spring to spring.

11. The Metropolis algorithm is a well-known approach to solving complex minimization (or maximization) problems, inspired by thermodynamics (Press, Teukolsky, Vetterling, & Flannery, 1992, p. 444). When a liquid is slowly cooled until it is solid, the resulting crystal is very orderly and has almost the minimum energy possible. Metropolis takes a function representing the energy of a system and applies simulated annealing solving the minimization problem by modeling the effect slow cooling has on the energy of the system. Metropolis does not suffer from the local minimum problem.

Figure 26. **(a) Hook's keyboard; (b) Metropolis I keyboard; (c) Metropolis II keyboard. From "Physics-based graphical keyboard design," by M. Hunter, S. Zhai, and B. A. Smith, 2000, *Extended Abstracts of the CHI 2000 Conference on Human Factors in Computing Systems.* Copyright 2000 by ACM. Reprinted with permission. From "The Metropolis keyboard: An exploration of quantitative techniques for virtual keyboard design," by S. Zhai, M. Hunter, and B. A. Smith, 2000, *Proceedings of the ACM Conference on User Interface Software and Technology–UIST 2000.* Copyright 2000 by ACM. Reprinted with permission.**



thumb typing, the DotNote keyboard fills most of the display with relatively large keys; however, this allows only half of the alphabet to appear at once. The most common letters appear on the default DotNote keyboard (Figure 28a) and a mode Shift key switches to the second keyboard arrangement, which contains the less common letters (Figure 28b). The arrangement of the keys in each soft keyboard is alphabetic. No published studies of the DotNote keyboard exist.

Figure 27. **Lewis keyboard. From "Development of a digram-based typing key layout for single-finger/stylus input," by J. R. Lewis, M. J. LaLomia, and P. J. Kennedy, 1999, *Proceedings of the Human Factors and Ergonomics Society 43rd Annual Meeting-HFES 99*. Copyright 1999 by the Human Factors and Ergonomics Society. Reprinted with permission.**

Lawrence Erlbaum Associates, Inc. does not have electronic rights to Figure 27. Please see the print version.

Figure 28. **DotNote soft keyboard: (a) the default key arrangement containing the more frequent letters; (b) the secondary key arrangement containing the less frequent letters.**

a)

				←
A	C	D	E	⇧
G	H	I	L	123
M	N	O	P	A↔Q
R	S	T	U	Space

b)

B	F	J	K	
Q	V	W	X	123
Y	Z	.	,	A↔Q
"	_			

4.3. Predictive Input Techniques

One early predictive input technology is the Reactive Keyboard (Darragh, 1988; Darragh & Witten, 1991; Darragh, Witten, & James, 1990). The Reactive Keyboard monitors what a user enters and presents text predictions that the user can choose from using the mouse. The predictions are generated by finding the longest matching substrings in the previously entered text. The Reactive Keyboard adapts to users' input and hence is not limited to a static set of words or phrases. No experimental results of text entry speed or accuracy are reported for the Reactive Keyboard. Other related work is hereby cited (Jakobsson, 1986; Masui & Nakayama, 1994; Raita & Teuhola, 1987).

POBox (Masui, 1998, 1999) is predictive input technology that allows users to enter part of a word and then search for similar words by spelling, pronunciation, or shape (for pictograph-based languages). It is not limited to alphabetic languages. POBox uses a static database coupled with another primary input technique, such as a soft keyboard or telephone keypad. Search results appear on the screen as the user types. A tap or key press selects the desired word. When embedded in a mobile phone, text entry via the multikey method yields a list of search results that the user scrolls through using a wheel on the side of the device.

Lewis (1999) and Lewis et al. (1999) experimented with a predictive soft keyboard technology for extremely limited screen sizes. Their system presents the user with keys for the six to eight most likely characters, and an "other" key revealing the rest of the alphabet. Lewis (1999) reported text-entry speeds for a Qwerty soft keyboard, his predictive keyboard, and handwriting (as a control condition) at 14 wpm, 6 wpm, and 22 wpm, respectively. The speed Lewis reported for soft keyboard entry is approximately half that reported by others (e.g., MacKenzie et al., 1994). Lewis observed that the uncertain arrangement of keys on the predictive keyboard significantly hindered performance.

5. CONCLUSIONS AND FUTURE WORK

There are many text entry methods available to designers of mobile systems, and without a doubt more are forthcoming. However, deciding which is best for an application is difficult, in part, because of the lack of publications giving empirically measured text entry speeds and accuracies. This article has brought together many of the techniques in use or under investigation in this exciting area in mobile computing. The result is a snapshot of the current state of the art in mobile text entry.

Some important modeling techniques have been presented and elaborated. Movement and language are omnipresent in human–computer interaction. The Fitts' Digram Model shows how Fitts' law and a language corpus can

work together in a priori analyses of design alternatives for stylus input on soft keyboards or single-finger input on small physical keyboards. However, further work is needed to refine this modeling technique—for example, in determining the correct coefficients for the Fitts' law model and in exploring and refining other aspects of the model, such as treatment of the space and punctuation characters or its sensitivity to changes in the language model.

In addition, we have examined many issues in methodology and evaluation and have identified factors, such as focus of attention, and whether one or two hands are used to manipulate the text entry device that impact user performance. Clearly, evaluation is critical, and it is by no means simple. A number of issues are particularly tricky, such as the measurement and treatment of errors and the types of tasks used in text entry studies. These and other topics are the subject of ongoing and future work.

NOTES

Acknowledgments. We thank Shumin Zhai, Barton Smith, and Michael Hunter for pointing out an error in the treatment of the Space character in our prediction model for soft keyboards. Their assistance—and persistence—in refining, improving, and extending our previous work is greatly appreciated.

Support. We gratefully acknowledge the support of the Natural Sciences and Engineering Research Council of Canada.

Authors' Present Addresses. Scott MacKenzie, Department of Computer Science, York University, Toronto, Ontario, Canada, M3J 1P3. E-mail: smackenzie@acm.org; William Soukoreff, Department of Computer Science, York University, Toronto, Ontario, Canada, M3J 1P3. E-mail: will@acm.org.

HCI Editorial Record. First manuscript received November 9, 2000. Accepted by Brad Myers. Final manuscript received June 1, 2001. — *Editor*

REFERENCES

Alsio, G., & Goldstein, M. (2000). Productivity prediction by extrapolation: Using workload memory as a predictor of target performance. *Behaviour & Information Technology, 19*(2), 87–96.

American Psychological Association. (1995). *Publication manual of the American Psychological Association* (4th ed.). Washington, DC: Author.

Bellman, T., & MacKenzie, I. S. (1998). A probabilistic character layout strategy for mobile text entry. *Proceedings of Graphics Interface 98*. Toronto: Canadian Information Processing Society.

Blickenstorfer, C. H. (1995, January). Graffiti: Wow!!!! *Pen Computing Magazine,* 30–31.

Bohan, M., Phipps, C. A., Chaparro, A., & Halcomb, C. G. (1999). A psychophysical comparison of two stylus-driven soft keyboards. *Proceedings of Graphics Interface 99*. Toronto: Canadian Information Processing Society.

Card, S. K., English, W. K., & Burr, B. J. (1978). Evaluation of mouse, rate-controlled isometric joystick, step keys, and text keys for text selection on a CRT. *Ergonomics, 21*, 601–613.

Card, S. K., Moran, T. P., & Newell, A. (1983). *The psychology of human–computer interaction.* Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Chang, L., & MacKenzie, I. S. (1994). A comparison of two handwriting recognizers for pen-based computers. *Proceedings of CASCON 94.* Toronto, Canada: IBM.

Damerau, F. J. (1964). A technique for computer detection and correction of spelling errors. *Communications of the ACM, 7*(3), 171–176.

Darragh, J. J. (1988). *Adaptive predictive text generation and the Reactive Keyboard.* Unpublished thesis, University of Calgary, Canada.

Darragh, J. J., & Witten, I. H. (1991). Adaptive predictive text generation and the reactive keyboard. *Interacting With Computers, 3*(1), 27–50.

Darragh, J. J., Witten, I. H., & James, M. L. (1990). The reactive keyboard: A predictive typing aid. *Computer, 23*(11), 41–49.

Devoe, D. B. (1967). Alternatives to handprinting in the manual entry of data. *IEEE Transactions on Human Factors in Engineering, HFE–8*, 21–32.

Dix, A., Finlay, J., Abowd, G., & Beale, R. (1998). *Human–computer interaction* (2nd ed.). London: Prentice Hall.

Enns, N., & MacKenzie, I. S. (1998). Touch-based remote control devices. *Extended Abstracts of the CHI 98 Conference on Human Factors in Computing Systems.* New York: ACM.

Fitts, P. M. (1954). The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology, 47*, 381–391.

Frankish, C., Hull, R., & Morgan, P. (1995). Recognition accuracy and user acceptance of pen interfaces. *Proceedings of the CHI 95 Conference on Human Factors in Computing Systems.* New York: ACM.

Fukumoto, M., & Suenaga, Y. (1994). FingeRing: A full-time wearable interface. *Proceedings of CHI 94 Conference on Human Factors in Computing Systems.* New York: ACM.

Gentner, D. R., Grudin, J. T., Larochelle, S., Norman, D. A., & Rumelhart, D. E. (1983). A glossary of terms including a classification of typing errors. In W. E. Cooper (Ed.), *Cognitive aspects of skilled typing* (pp. 39–44). New York: Springer-Verlag.

Gibbs, M. (1993). Handwriting recognition: A comprehensive comparison. *Pen,* (March/April), 31–35.

Goldberg, D., & Richardson, C. (1993). Touch-typing with a stylus. *Proceedings of the INTERCHI 93 Conference on Human Factors in Computing Systems.* New York: ACM.

Goldstein, M., Book, R., Alsio, G., & Tessa, S. (1999). Non-keyboard Qwerty touch typing: A portable input interface for the mobile user. *Proceedings of the CHI 99 Conference on Human Factors in Computing Systems.* New York: ACM.

Gopher, D., & Raij, D. (1988). Typing with a two-hand chord keyboard: Will the Qwerty become obsolete? *IEEE Transactions of Systems, Man, and Cybernetics, 18*, 601–609.

Grinter, R. E., & Eldridge, M. A. (2001). y do tngrs luv 2 txt msg? *Proceedings of the ECSCW 2001 European Conference on Computer Supported Collaborative Work.* Dordrecht, The Netherlands: Kluwer Academic.

Grover, D. L., King, M. T., & Kuschler, C. A. (1998). *Reduced keyboard disambiguating computer.* Seattle, WA: Tegic Communications.

Hancock, P. A., & Newell, K. M. (1985). The movement speed–accuracy relationship in space-time. In H. Heuer, U. Kleinbeck, & K.-H. Schmidt (Eds.), *Motor behavior: Programming, control and acquisition* (pp. 153–188). New York: Springer-Verlag.

Hick, W. E. (1952). On the rate of gain of information. *Quarterly Journal of Experimental Psychology, 4,* 11–36.

Hjelm, J., Tan, C. L., Fabry, L., Fanchon, T., & Reichert, F. (1996). Building the UMTS user interface. *Advanced Communications Technologies and Services (ACTS) Mobile Communications Summit* (Vol. 2, pp. 687–692). Brussels, Belgium: Acts European Commission.

Hunter, M., Zhai, S., & Smith, B. A. (2000). Physics-based graphical keyboard design. *Extended Abstracts of the CHI 2000 Conference on Human Factors in Computing Systems.* New York: ACM.

Hyman, R. (1953). Stimulus information as a determinent of reaction time. *Journal of Experimental Psychology, 45,* 188–196.

Isokoski, P. (1999). *A minimal device-independent text input method.* Unpublished thesis, University of Tampere, Finland.

Jakobsson, M. (1986). Autocompletion in full text transcription entry: A method for humanized input. *Proceedings of the CHI 86 Conference on Human Factors in Computing Systems.* New York: ACM.

Kay, A., & Goldberg, A. (1977, March). Personal dynamic media. *Computer,* 31–41.

LaLomia, M. J. (1994). User acceptance of handwritten recognition accuracy. *Companion Proceedings of the CHI 94 Conference on Human Factors in Computing Systems.* New York: ACM.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics–Doklady, 10,* 707–710.

Lewis, J. R. (1999). Input rates and user preference for three small-screen input methods: Standard keyboard, predictive keyboard, and handwriting. *Proceedings of the Human Factors and Ergonomics Society 43rd Annual Meeting. Santa Monica, CA: Human Factors and Ergonomics Society.*

Lewis, J. R., Allard, D. J., & Hudson, H. D. (1999). Predictive keyboard design study: Effects of word populations, number of displayed letters, and number of transitional probability tables. *Proceedings of the Human Factors and Ergonomics Society 43rd Annual Meeting.* Santa Monica, CA: Human Factors and Ergonomics Society.

Lewis, J. R., LaLomia, M. J., & Kennedy, P. J. (1999a). Development of a digram-based typing key layout for single-finger/stylus input. *Proceedings of the Human Factors and Ergonomics Society 43rd Annual Meeting.* Santa Monica, CA: Human Factors and Ergonomics Society.

Lewis, J. R., LaLomia, M. J., & Kennedy, P. J. (1999b). Evaluation of typing key layouts for stylus input. *Proceedings of the Human Factors and Ergonomics Society 43rd Annual Meeting. Santa Monica, CA: Human Factors and Ergonomics Society.*

MacKenzie, I. S. (1992). Fitts' law as a research and design tool in human–computer interaction. *Human–Computer Interaction, 7,* 91–139.

MacKenzie, I. S., & Chang, L. (1999). A performance comparison of two handwriting recognizers. *Interacting with Computers, 11,* 283–297.

MacKenzie, I. S., Nonnecke, R. B., McQueen, J. C., Riddersma, S., & Meltz, M. (1994). A comparison of three methods of character entry on pen-based computers. *Proceedings of the Human Factors Society 38th Annual Meeting.* Santa Monica, CA: Human Factors and Ergonomics Society.

MacKenzie, I. S., Nonnecke, R. B., Riddersma, S., McQueen, C., & Meltz, M. (1994). Alphanumeric entry on pen-based computers. *International Journal of Human–Computer Studies, 41*, 775–792.

MacKenzie, I. S., Sellen, A., & Buxton, W. (1991). A comparison of input devices in elemental pointing and dragging tasks. *Proceedings of the CHI 91 Conference on Human Factors in Computing Systems.* New York: ACM.

MacKenzie, I. S., & Zhang, S. X. (1997). The immediate usability of Graffiti. *Proceedings of Graphics Interface 97.* Toronto: Canadian Information Processing Society.

MacKenzie, I. S., & Zhang, S. X. (1999). The design and evaluation of a high-performance soft keyboard. *Proceedings of the CHI 99 Conference on Human Factors in Computing Systems.* New York: ACM.

MacKenzie, I. S., & Zhang, S. X. (2000). *An investigation of the novice experience with soft keyboards*. Manuscript submitted for publication.

MacKenzie, I. S., Zhang, S. X., & Soukoreff, R. W. (1999). Text entry using soft keyboards. *Behaviour & Information Technology, 18*, 235–244.

Mankoff, J., & Abowd, G. A. (1998). Cirrin: A word-level unistroke keyboard for pen input. *Proceedings of the UIST 98 Symposium on User Interface Software and Technology.* New York: ACM.

Martin, D. W. (1996). *Doing psychology experiments* (4th ed.). Pacific Grove, CA: Brooks/Cole.

Masui, T. (1998). An efficient text input method for pen-based computers. *Proceedings of the CHI 98 Conference on Human Factors in Computing Systems.* New York: ACM.

Masui, T. (1999). POBox: An efficient text input method for handheld and ubiquitous computers. *Proceedings of the HUC 99 International Symposium on Handheld and Ubiquitous Computing.* Berline, Germany: Springer-Velag.

Masui, T., & Nakayama, K. (1994). Repeat and predict: Two keys to efficient text editing. *Proceedings of the CHI 94 Conference on Human Factors in Computing Systems.* New York: ACM.

Matias, E., MacKenzie, I. S., & Buxton, W. (1993). Half-Qwerty: A one-handed keyboard facilitating skill transfer from Qwerty. *Proceedings of the INTERCHI 93 Conference on Human Factors in Computing Systems.* New York: ACM.

Matias, E., MacKenzie, I. S., & Buxton, W. (1994). Half-Qwerty: Typing with one hand using your two-handed skills. *Companion Proceedings of the CHI 94 Conference on Human Factors in Computing Systems.* New York: ACM.

Matias, E., MacKenzie, I. S., & Buxton, W. (1996a). One-handed touch typing on a Qwerty keyboard. *Human–Computer Interaction, 11*, 1–27.

Matias, E., MacKenzie, I. S., & Buxton, W. (1996b). A wearable computer for use in microgravity space and other non-desktop environments. *Companion Proceedings of the CHI 96 Conference on Human Factors in Computing Systems.* New York: ACM.

Mayzner, M. S., & Tresselt, M. E. (1965). Table of single-letter and digram frequency counts for various word-length and letter-position combinations. *Psychonomic Monograph Supplements, 1*(2), 13–32.

McMulkin, M. (1992). Description and prediction of long-term learning of a keyboarding task. *Proceedings of the Human Factors Society 36th Annual Meeting.* Santa Monica, CA: Human Factors Society.

Pachella, R. G., & Pew, R. W. (1968). Speed–accuracy tradeoff in reaction-time: Effect of discrete criterion times. *Journal of Experimental Psychology, 76*(1), 19–24.

Perlin, K. (1998). Quikwriting: Continuous stylus-based text entry. *Proceedings of the UIST 98 Symposium on User Interface Software and Technology.* New York: ACM.

Pew, R. W. (1969). The speed–accuracy operation characteristic. In W. G. Koster (Ed.), *Attention and performance II* (pp. 16–26). Amsterdam: North-Holland.

Poon, A., Weber, K., & Cass, T. (1995). Scribbler: A tool for searching digital ink. *Companion Proceedings of the CHI 95 Conference on Human Factors in Computing Systems.* New York: ACM.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (1992). *Numerical recipes in C: The art of scientific computing* (2nd ed.). Cambridge, England: Cambridge University Press.

Raita, T., & Teuhola, J. (1987). Predictive text compression by hashing. *Proceedings of the Tenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* New York: ACM.

Rau, H., & Skiena, S. (1994). Dialing for documents: An experiment in information theory. *Proceedings of the UIST 94 Symposium on User Interface Software and Technology.* New York: ACM.

Sacher, H. (1998, September/October). Interactions in Chinese: Designing interfaces for Asian languages. *Interactions,* 28–38.

Santos, P. J., Baltzer, A. J., Badre, A. N., Henneman, R. L., & Miller, M. S. (1992). On handwriting recognition system performance: Some experimental results. *Proceedings of the Human Factors Society 36th Annual Meeting.* Santa Monica, CA: Human Factors and Ergonomics Society.

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

Shneiderman, B. (2000, September). The limits of speech recognition. *Communications of the ACM,* 63–65.

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

Soukoreff, R. W., & MacKenzie, I. S. (1995). Theoretical upper and lower bounds on typing speeds using a stylus and soft keyboard. *Behaviour & Information Technology, 14,* 370–379.

Soukoreff, R. W., & MacKenzie, I. S. (2001). Measuring errors in text entry tasks: An application of the Levenshtein string distance statistic. *Companion Proceedings of the CHI 2001 Conference on Human Factors in Computing Systems.* New York: ACM.

Sugimoto, M., & Takahashi, K. (1996). SHK: Single hand key card for mobile devices. In *Companion Proceedings of the CHI 96 Conference on Human Factors in Computing Systems.* New York: ACM.

Suhm, B., Myers, B., & Waibel, A. (1999). Model-based and empirical evaluation of multimodal interactive error correction. *Proceedings of the CHI 99 Conference on Human Factors in Computing Systems.* New York: ACM.

Swensson, R. G. (1972). The elusive tradeoff: Speed vs. accuracy in visual discrimination tasks. *Perception and Psychophysics, 12*(1A), 16–32.

Tappert, C. C., Suen, C. Y., & Wakahara, T. (1990). The state of the art in on-line handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 12*, 787–808.

Trudeau, G. B. (1996). *Flashbacks: Twenty-five years of Doonesbury*. Kansas City, MO: Andrews McMeel.

Underwood, B. J., & Schulz, R. W. (1960). *Meaningfulness and verbal learning*: Philadelphia: Lippincott.

Van Cott, H. P., & Kinkade, R. G. (Eds.). (1972). *Human engineering guide to equipment design*. Washington, DC: U.S. Government Printing Office.

Venolia, D., & Neiberg, F. (1994). T-Cube: A fast, self-disclosing pen-based alphabet. *Proceedings of CHI 94 Conference on Human Factors in Computing Systems*. New York: ACM.

Walker, N., & Catrambone, R. (1993). Aggregation bias and the use of regression in evaluation models of human performance. *Human Factors, 35*, 397–411.

Ward, D. J., Blackwell, A. F., & MacKay, D. J. C. (2000). Dasher: A data entry interface using continuous gestures and language models. *Proceedings of the UIST 2000 Symposium on User Interface and Software Technology*. New York: ACM.

Ward, D. J., Blackwell, A. F., & MacKay, D. J. C. (2002). Dasher: A gesture-driven data entry interface for mobile computing. *Human–Computer Interaction,* 17, 199–228.

Wickelgren, W. A. (1977). Speed–accuracy tradeoff and information processing dynamics. *Acta Psychologica, 41*, 67–85.

Wolf, C. G., Glasser, A. R., & Fujisaki, T. (1991). An evaluation of recognition accuracy for discrete and run-on writing. *Proceedings of the Human Factors Society 35th Annual Meeting*. Santa Monica, CA: Human Factors and Ergonomics Society.

Zhai, S., Hunter, M., & Smith, B. A. (2000). The Metropolis keyboard: An exploration of quantitative techniques for virtual keyboard design. *Proceedings of the UIST 2000 Symposium on User Interface Software and Technology*. New York: ACM.

Zhang, S. X. (1998). *A high performance soft keyboard for mobile systems*. Unpublished thesis, University of Guelph, Ontario, Canada.

Copyright of Human-Computer Interaction is the property of Lawrence Erlbaum Associates and its content may not be copied or emailed to multiple sites or posted to a listserv without the copyright holder's express written permission. However, users may print, download, or email articles for individual use.