Human–Computer Interaction

Publication details, including instructions for
authors and subscription information:
http://www.tandfonline.com/loi/hhci20

Performance Optimization of Virtual Keyboards

Shumin Zhai , Michael Hunter & Barton A. Smith
Published online: 22 Jun 2011.

PLEASE SCROLL DOWN FOR ARTICLE

Performance Optimization of Virtual Keyboards

**Shumin Zhai, Michael Hunter,
and Barton A. Smith**
IBM Almaden Research Center

ABSTRACT

Text entry has been a bottleneck of nontraditional computing devices. One of the promising methods is the virtual keyboard for touch screens. Correcting previous estimates on virtual keyboard efficiency in the literature, we estimated the potential performance of the existing Qwerty, FITALY, and OPTI designs of virtual keyboards to be in the neighborhood of 28, 36, and 38 words per minute (wpm), respectively. This article presents 2 quantitative design techniques to search for virtual keyboard layouts. The first technique simulated the dynamics of a keyboard with digraph springs between keys, which produced a Hooke keyboard with 41.6 wpm movement efficiency. The second technique used a Metropolis random walk algorithm guided by a "Fitts-digraph energy" objective function that quantifies the movement efficiency of a virtual keyboard. This method produced various Metropolis keyboards with different shapes and

Shumin Zhai is a human–computer interaction researcher with an interest in inventing and analyzing interaction methods and devices based on human performance insights and experimentation; he is a Research Staff Member in the User Sciences and Experience Research Department of the IBM Almaden Research Center. **Michael Hunter** is a graduate student of Computer Science at Brigham Young University; he is interested in designing graphical and haptic user interfaces. **Barton A. Smith** is an experimental scientist with an interest in machines, people, and society; he is manager of the Human Interface Research Group at the IBM Almaden Research Center.

CONTENTS

structures with approximately 42.5 wpm movement efficiency, which was 50% higher than Qwerty and 10% higher than OPTI. With a small reduction (41.16 wpm) of movement efficiency, we introduced 2 more design objectives that produced the ATOMIK layout. One was alphabetical tuning that placed the keys with a tendency from A to Z so a novice user could more easily locate the keys. The other was word connectivity enhancement so the most frequent words were easier to find, remember, and type.

1. INTRODUCTION

Pervasive devices have come to the forefront in computer technology. Small handheld devices such as personal digital assistants (PDAs), pagers, and mobile

phones, as well as larger scale devices such as tablet computers and electronic whiteboards, now play an increasingly more central role in human–information interaction. This general trend is rapidly freeing us from the confines of our laptop or desktop computers and leading us to a future of pervasive computing.

Despite this favorable trend in pervasive computing, certain obstacles stand in the way of developing efficient applications on these devices. An obvious problem relates to text input. For example, a recent study by McClard and Somers (2000) clearly demonstrated the value of tablet computers in home environments. However, the lack of efficient text input techniques in these tablet computers made many common applications, such as chat, e-mail, or even entering a URL very difficult.

Text entry is also problematic for PDAs and other handheld devices. Currently, text input on these devices can be achieved through reduced physical keyboards, handwriting recognition, voice recognition, and virtual keyboards, but each has critical usability shortcomings. We briefly describe several representative methods. See MacKenzie and Soukoreff (2002) for a more detailed survey of these methods.

Physical Keyboards.   There are two ways to reduce the size of physical keyboards. One is to shrink the size of each key. This is commonly seen in electronic dictionaries. Typing on these keyboards is slow and difficult due to their reduced size. The other method is to use the number pads in telephones, whereby each number corresponds to multiple letters. The ambiguity of multiple possible letters is commonly resolved by the number of consecutive taps, or by lexical models.

Handwriting.   Reducing error rate has been the major goal in handwriting recognition. However, the ultimate bottleneck of handwriting input lies in the human handwriting speed limit. It is very difficult to write legibly at a high speed.

Voice Recognition.   Speech has been expected to be a compelling alternative to typing. Despite the progress made in speech recognition technology, however, a recent study by Karat, Halverson, Horn, and Karat (1999) showed that the effective speed of text entry by continuous speech recognition was still far lower than that of the keyboard (13.6 vs. 32.5 corrected words per minute [wpm] for transcription and 7.8 vs. 19.0 corrected wpm for composition). Furthermore, the study also revealed many human-factors issues that had not been well understood. For example, many users found it "harder to talk and think than type and think" and considered the keyboard to be more "natural" than speech for text entry.

There have also been other continuous-gesture-based text methods. A recent example is Dasher (Ward, Blackwell, & MacKay, 2000), which uses continuous mouse movement to pass through traces of letters laid out by a predictive language model. Using such a technique is a novel and intriguing experience, but the primary drawback is that the user has to continuously recognize the dynamically arranged letters. The visual recognition task may limit the eventual performance of text entry with such a method.

This article focuses on the design of virtual keyboards. Such a keyboard displays letters and numbers on a touch sensitive screen or surface. To input text, the user presses keys with a finger or stylus. Such a keyboard can be scaled to fit computing devices with varying sizes, particularly small handheld devices. One central issue, however, is the layout of the keys in these keyboards. Due to developers' and users' existing knowledge, the Qwerty layout used in most physical keyboards today has the momentum to become the most likely choice. In fact, some PDA products, such as the Palm™ Pilot, have already used the Qwerty as their virtual keyboard layout.

Unfortunately, the Qwerty layout (see Figure 1) designed by Christopher L. Sholes, Carlos Glidden, and Samuel W. Soule in 1868 is a poor choice for virtual keyboards. This is because the Qwerty keyboard was so arranged that many adjacent letter pairs (digraphs) appear on the opposite sides of the keyboard. The main purpose of this arrangement was to minimize mechanical jamming (Cooper, 1983; Yamada, 1980). Accidentally, this design also facilitates the frequent alternation of the left and right hand, which is a key premise to rapid touch typing with two hands that was discovered many years after the typewriter was invented. Partially because the Qwerty design scores well in alternation frequency, various attempts to replace Qwerty with more efficient layouts, such as the Dvorak simplified keyboard (Dvorak, Merrick, Dealey, & Ford, 1936), have not prevailed. The performance gain with these newer designs (around 15%) has not been substantial enough to justify the cost of retraining the great number of Qwerty users (Cooper, 1983; Norman & Fisher, 1982; Yamada, 1980). However, on a virtual keyboard, the polarizing common digraphs in Qwerty mean that the stylus has to move back and forth more frequently and over greater distances than necessary. The key to a good virtual keyboard is exactly opposite to the idea behind Qwerty. Common digraph letters should be close to each other so the hand does not have to travel much. The movement distance concern also points to another problem of Qwerty as a virtual keyboard layout, it is elongated horizontally, which statistically increases the stylus movement distances. In fact, the human performance effect of relative distances between the letters can be modeled by a simple movement equation—the Fitts' law.

There are additional reasons to thoroughly study virtual keyboard layouts at this point in user interface history. First, it is not too late to form a new layout

Figure 1. **The Qwerty layout designed by Sholes, Glidden, and Soule in 1868.**
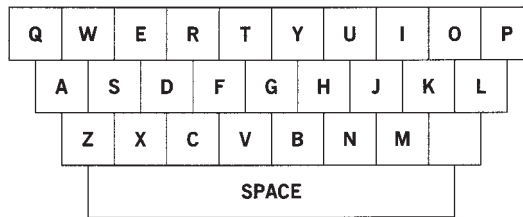
standard for the virtual keyboard, due to the relatively small number of people using virtual keyboards today. Second, the 10-finger touch typing skills on a physical keyboard do not necessarily transfer to on-screen stylus tapping on the same layout (Zhang, 1998). The perceptual, memory, and motor behavior of using a virtual keyboard is sufficiently different from that of a physical keyboard to justify a different design.

2. PERFORMANCE MODELING OF VIRTUAL KEYBOARDS

To minimize finger movement on a virtual keyboard, two factors must be taken into account. One is the transitional frequencies from one letter to another in a given language (digraph statistics), and the other is the relative distances between keys. The goal should be to arrange the letters so that the statistical total travel distance is the shortest when tapping on such a keyboard. This means that the most frequent keys should be located in the center of the keyboard and the frequently connected letters (such as *T* and *H*) should be closer to each other than the less frequently connected letters.

2.1. The Fitts-Digraph Model of Virtual Keyboards

MacKenzie and colleagues (MacKenzie & Zhang, 1999; Soukoreff & MacKenzie, 1995) were the first to use a quantitative approach to model virtual keyboard performance. Their model predicts user performance by summing the Fitts' law movement times (MTs) between all digraphs, weighted by the frequencies of occurrence of the digraphs. The use of Fitts' law made it possible to estimate performance in absolute terms, giving us a comparison to speed we are familiar with, such as 60 wpm for a good touch typist.

According to Fitts' law (Figure 2), the time to move the tapping stylus from one key i to another j for a given distance (D_{ij}) and key size (W_j) is[1]

---

1. i and j here represent any pair of keys from A to Z and the space key.

Figure 2. **The average movement time can be predicted by Fitts' law.**

$$MT = a + b\ Log_2(D_{ij}/W_j + 1)$$
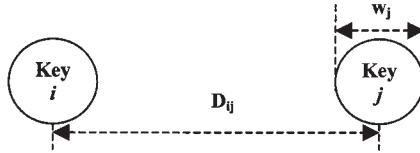
$$MT = a + b\log_2\left(\frac{D_{ij}}{W_j} + 1\right),\tag{1}$$

where *a* and *b* are empirically determined coefficients. To be able to make comparisons to the results in the literature (e.g., MacKenzie & Zhang, 1999), we choose $a = 0$, $b = 1 / 4.9$. In other words, we consider the Fitts' index of performance (IP; Fitts, 1954) to be 4.9 bits per second (bps). We return to the choice of this parameter later.

If the frequency of letter *j* to follow letter *i* (digraph *I–j*) among all digraphs is P_{ij}, then the mean time in seconds for typing a character is:

$$t = \sum_{i=1}^{27}\sum_{j=1}^{27}\frac{P_{ij}}{IP}\left[log_2\left(\frac{D_{ij}}{W_j} + 1\right)\right],\tag{2}$$

Assuming five characters per word (including space key), this equation allows us to calculate tapping speed in wpm ($60 / 5\ t$).

Note that a special case has to be made for Equation 2 when $i = j$. This is when the user taps on the same key successively (e.g., *oo* as in *look*). In this case, the second term in Equation 1 is 0 but *a* is set at .127 sec. Previous researchers (MacKenzie & Zhang, 1999; Zhang, 1998) have used both .127 and .135 sec. We chose .127 because it was closer to what we measured (7.8 repeats on the same location). The influence of this number is small, however, due to the low frequency of such cases.

It should be emphasized that Equation 2 only estimates the movement efficiency of tapping on a virtual keyboard. An expert user could possibly achieve this efficiency. A novice or intermediate user has to visually search for the destination key before tapping on it. In that sense, Equation 2 only predicts the potential upper bound of a user's performance (Soukoreff & MacKenzie, 1995). However, the Fitts' law coefficient in the model is based on average human tapping performance. Some users, therefore, could surpass this "upper bound."

2.2. Digraph Frequency

The digraph frequency P_{ij} in Equation 2 is numerically calculated by the ratio between the number of i–j digraphs and the total number of digraphs in an English text corpus. One commonly used digraph table was made by Mayzner and Tresselt, extracted from an English text corpus of 87,296 characters (Mayzner & Tresselt, 1965). There are two shortcomings to this digraph table. First, it is not clear whether their corpus still accurately reflects current English word use, in particular the sort of language used in digital media. Second, their corpus was limited to words with three to seven letters. This restriction eliminated many of the most frequently used words (e.g., *I*, *in*, *on*, *is*, *at*, *to*, *of*, *it*, and *if*).

We hence constructed two new text corpora and two digraph tables. One was sampled from online news articles from sources such as the *New York Times,* the *LA Times* and the *San Jose Mercury News.* The articles covered a range of topics from technical to social–political. The size of the corpus was 101,468 characters (without counting spaces between the words). The second 1,364,497-character corpus was gathered from logs of six online chat rooms. The names of the arbitrarily selected chat rooms were Teen, Atheism, ChristianDebate, Myecamp, CityoftheGreats, and MaisonlkkoguRPG. The text in the chat room corpus consisted of very informal conversations. Most input strings were less than 80 characters, and many were not complete sentences. They were frequently a one- or two-word answer to a question or comment about a previous statement. Capitalization at the beginning and periods at the end of sentences were frequently missing. We included only those records that appeared to be typed by a person. Computer-generated headers and other text were deleted.

We collected two corpora because of the informal style of English used in applications such as e-mail, instant messaging, and chat, for which a virtual keyboard will likely be used. We were concerned that a keyboard optimized for standard formal English may not be optimal for informal electronic writing. As we see later in this article, however, the difference in language style does not significantly alter the movement efficiency of a virtual keyboard, possibly because the phonology of the English language, which determines the digraph distribution, does not change significantly with the formality of the language.

The set of characters tabulated is also different from the digraph table of Mayzner and Tresselt (1965). Their table was concerned only with alphabetical characters. In fact, designers had to reconstruct the Mayzner and Tresselt table with an added space key by inference (e.g., Soukoreff & MacKenzie, 1995). We collected and tabulated 128 × 128 symbols, including many nonalphabetic characters. Because of space limitations, we could provide only the core portions of the digraph tables in the Appendix. They included the space, Roman alphabet, and nine most frequent punctuation mark characters

in each of the corpora.[2] In tabulating these data, heading text generated by the IRC program was first deleted, then uppercase Roman alphabet characters were converted to lowercase before the digraphs were counted. Figures 3 and 4 illustrate the most frequent symbols in the two corpora.

To be able to compare with data reported in the literature, we continued to use the Mayzner and Tresselt table in evaluating existing keyboard designs. We used all three tables in designing the Metropolis keyboards presented later in this article.

2.3. Existing Layouts and Their Movement Efficiency Estimation

To put in context our proposed virtual keyboard design (presented in a later section), this section details various existing layouts of virtual keyboards and applies the Fitts-digraph model (Equation 2) to estimate the movement efficiency of these layouts. This is necessary for two reasons. First, other than informal arguments and promotional data, some of the existing layouts have never been scientifically evaluated. Second, previously published estimates of Qwerty and OPTI keyboards (MacKenzie & Zhang, 1999; Zhang, 1998) in the literature need to be corrected.

Qwerty

Applying Equation 2 to evaluate the movement efficiency of the Qwerty keyboard is straightforward, except with respect to the treatment of the Space key. The Space key in the Qwerty layout has a much greater length than the rest of the keys. The Fitts' law distance between a character key and the Space key varies depending on what point of the Space key is tapped. MacKenzie and Zhang (MacKenzie & Zhang, 1999; Zhang, 1998) used a "suboptimal" model to handle the Space key, which divided the Space key into multiple segments; each was equal in length to a regular character key. For each character–space–character "trigraph," they chose the segment of the Space key that yielded the shortest total distance of character–space–character path. Then they calculated the two Fitts' law times of character–space and of space–character according to the distances along that path. Finally, they summed the Fitts' tapping times of all of character–space–character trigraphs, weighted by the probability of each trigraph occurrence. Using this approach, they estimated the Qwerty movement efficiency of 43.2 wpm.

---

2. We will provide the complete tables to researchers upon request.

Figure 3. **Letter frequency in the chat room corpus.**
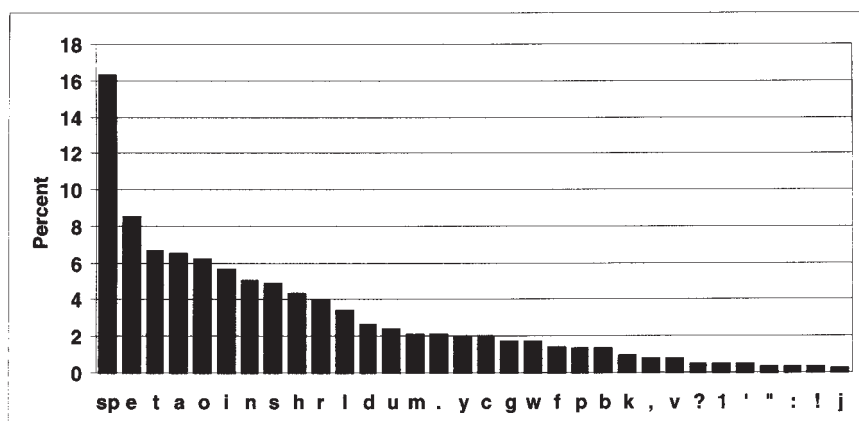
Figure 4. **Letter frequency in the news corpus.**
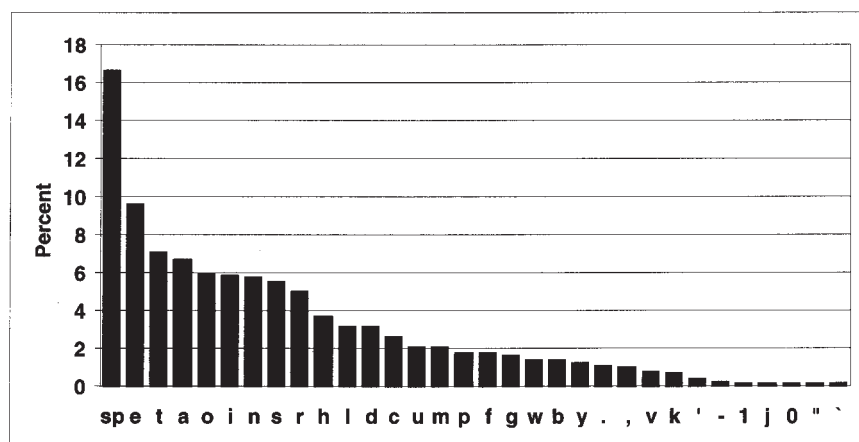
Following the same methodology, we could not replicate their result. On close examination, we found a subtle error in MacKenzie and Zhang's calculation of the probability of each character–space–character trigraph. Taking the combination of i–space–j as an example, they incorrectly used $P_{i-space} \times P_{space-j}$ to calculate the probability of the path $P_{i-space-j}$. Note that $P_{space-j}$ is the probability (or frequency) of the transition at any given tapping to be from the space key to the J key. It is not the *conditional* probability of $P_{space-j/space} \left(P_{space-j} / P_{space} \right)$

needed to calculate the probability of two serial events. The correct calculation should be:

$$P_{i-space-j} = P_{i-space} \times \ P_{space-j/space} = P_{i-space} \times \ P_{space-j} / P_{space} \qquad (3)$$

where $P_{i-space}$ is the probability of i–space digraph at any given tapping, $P_{space-j}$ is the probability of space–j at any given tapping, $P_{space-j/space}$ is the conditional probability of space–j, given the last tapped key is space, and P_{space} is the probability (frequency) of the Space key.

Using this corrected $P_{space-j/space}$ calculation but following the same "suboptimal" methodology as in (Zhang, 1998), we found the movement efficiency of Qwerty layout to be 30.5 wpm.

To ensure the correctness of our estimation, we also applied two much simpler methods, one conservative and one liberal. Both involved only digraphs and avoided character–space–character trigraphs. By the conservative method, character-to-space distance was always measured to the center of the Space key. Obviously the result of this should be lower than the estimation of suboptimal model. Indeed, the speed calculated by this method was 27.6 wpm. By the liberal method, the distance between the Space key and any character key was always measured along the shortest (vertical) line from the character to the Space key. Due to the "free warping" effect—the stylus goes into the Space key from one point and comes out from another point of the Space key without taking any time—this should produce a higher estimate than the suboptimal model. Indeed, we found the tapping speed to be 31.77 wpm with this method.

In conclusion, the movement efficiency of a Qwerty keyboard is about 30 wpm, assuming the user always taps on the portion of the Space key that minimizes the character–space–character total path. If the user does not plan one key ahead, the movement efficiency would be about 28 wpm.

Square Alphabetic

We have pointed out that Qwerty is worse than a random design as a virtual keyboard due to the polarization of common digraphs and its elongated shape. As a comparison to Qwerty, we evaluated a design that sequentially lays out the alphabetical letters in a 5×6 (column × row) grid (Figure 5). In fact, this was one of the layouts suggested by Lewis, Kennedy, and LaLomia (1999). Although this "design" does not consider digraph frequency distribution at all, its speed is 33.45 wpm, still faster than Qwerty. Note that such a result is based on the conservative assumption that the user also taps the middle of the spacebar at the bottom of the keyboard. Because the Space key is the most frequent key used, an obvious improvement is a 6×5 (column × row) design, as in Figure 6.

Figure 5. **A square alphabetic layout.**

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

Figure 6. **An improved alphabetic layout.**

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

MacKenzie–Zhang OPTI

MacKenzie and Zhang (MacKenzie & Zhang, 1999; Zhang, 1998) designed a new, optimized layout, dubbed OPTI (Figure 7). They first placed the 10 most frequent letters in the center of the keyboard. Then, assigning the 10 most frequent digraphs to the top 10 keys, they placed the remaining letters. The placement was all done by trial and error. They later made a further improved 5×6 layout, shown in Figure 8. For convenience, we call the 5×6 layout OPTI II in this article.

There are four space keys in both OPTI keyboards, evenly distributed in the layout. The user is free to choose any one of them. The optimal choice depends on both the preceding and following key to the Space key. For example, for the sequence of *M*–space–*V* (Figure 8), the upper right Space key is the best choice. However, the upper right space key is not the optimal choice if the tapping sequence is *M*–space–*Y*. In practice, the use of the optimal Space key ranged from 38% to 47%, depending on the user's experience (MacKenzie & Zhang, 1999).

Assuming optimal choice of Space keys, MacKenzie and Zhang (MacKenzie & Zhang, 1999; Zhang, 1998) predicted 58.2 wpm movement efficiency of the OPTI keyboard and 59.4 wpm on the OPTI II layout. These were surpris-

Figure 7. **MacKenzie's and Zhang's OPTI layout.**

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

Figure 8. **The improved OPTI layout in a 5 × 6 layout (OPTI II).**

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

ingly high performance scores that we could not replicate. As in their Qwerty estimation, MacKenzie and Zhang (MacKenzie & Zhang, 1999; Zhang, 1998) used the character–space–character trigraph approach to handle the multiple Space keys. They made the same probability miscalculation of the trigraphs on the OPTIs as they did on the Qwerty.

We recalculated the movement efficiency of the OPTI II keyboard. The first approach was the same as that of MacKenzie and Zhang, except we used the corrected conditional probability in calculating trigraph probabilities. Our result for the OPTI II is 40.3 wpm. This result was based on the assumption that the user *always* used the optimal Space key.

Our second approach took the nonoptimal Space keys into account. We assumed that the user would make use of the optimal Space key 50% of time, which was still higher than the highest actual rate measured (MacKenzie & Zhang, 1999). For the rest of the time, the average distance from the character key to the three nonoptimal Space keys was used. Using this approach, we found the OPTI II movement efficiency to be 36 wpm.

In conclusion, the OPTI II movement efficiency should be between 36 and 40.3 wpm, depending on the optimality of the Space key choice. If we take 38 wpm as a fair (but optimistic) estimate, this is a 35% improvement over Qwerty (28 wpm).

MacKenzie and Zhang conducted an experiment to investigate how quickly users could reach the predicted movement efficiency. In their test, participants reached 44.3 wpm after 20 sessions of text entry, each for 45 min, on the OPTI design (MacKenzie & Zhang, 1999). This is higher than our predicted performance of 38 wpm. We think this disparity is due to the low value of Fitts' law IP used in Equation 2. Originated in MacKenzie, Sellen, and Buxton (1991), 4.9 bps might be an overly conservative estimate of human tapping performance.[3] For two adjacent keys (1 bit), 4.9 bps means 204 ms per tap. This is much longer than what we measured (average 160 ms). The rate of IP reported in the literature for tapping varied widely. For example, Fitts' original report was 10.6 bps (Fitts, 1954). This rate dropped to 8.2 bps after adjusting for the effective width and for the Shannon–MacKenzie formulation (MacKenzie, 1992), but it is still much higher than 4.9 bps. In this article we continue to use 4.9 bps as the default value of IP for two reasons. First, it is on the conservative side. Second, results based on this assumption can be compared with data from previous studies. One should be aware, however, that all predicted performance scores in this article can be proportionally scaled according to the IP rate. For example, if we use 6 bps instead of 4.9 bps, the OPTI II movement efficiency would be $38 \times 6/4.9 = 46.5$ wpm. Figure 9 lists our movement efficiency estimates of various layouts when IP is 4.9, 6, and 8 bps.

FITALY

The FITALY keyboard (Figure 10) is a commercial product by Textware™ Solutions. The design rationale behind this layout included center placement of more frequent keys, dual double-sized Space keys, and the consideration of digraph frequencies (Textware Solutions, 1998).

In a loosely controlled contest (self-reporting with a witness, best performer rewarded with a prize), Textware Solutions collected 34 entries of text entry speed on a Palm Pilot PDA, with 19 contestants using FITALY, 9 using Graffiti, and 6 using Qwerty. FITALY received the highest average score (44.4 wpm), followed by Qwerty keyboard and Graffiti handwriting (both 28.2 wpm; Textware Solutions, 1998). Note that these scores were collected from motivated contestants.

Applying the Fitts-digraph movement efficiency model (Equation 2), we did a formal estimation of the FITALY layout. In our calculation, the two double-sized Space keys were treated differently from other regular keys. First, the

---

3. For this reason, Soukoreff and MacKenzie (1995) also duplicated their estimation with 14 bps.

Figure 9. **Summary of movement efficiency (in words per minute [wpm]) of virtual keyboards at different Fitts' law Index of Performance (IP) levels.**

Layouts	IP = 4.9 bits/sec	IP = 6.0 bits/sec	IP = 8.0 bits/sec	Estimation Bias
Qwerty	28	34.3	45.7	Slightly conservative[a]
5 × 6 Alphabetic	33.5	41.02	54.7	Slightly conservative[a]
OPTI II (MacKenzie–Zhang)	38	46.5	62.0	Liberal[b]
Fitaly (Textware Solutions)	36	44.1	58.8	Liberal[c]
Lewis–Kennedy–LaLomia	37.1	45.4	60.6	Slightly conservative[a]
Hooke (Zhai–Hunter–Smith)	41.6	50.9	67.9	Slightly conservative[d]
Metropolis (Zhai–Hunter–Smith)	43.1	52.1	69.4	Slightly conservative[d]
ATOMIK (Zhai–Hunter–Smith)	41.2	50.4	67.2	Slightly conservative[d]

[a]Assuming the user always taps the center of the long Space bar. All calculations in this table are based on Mayzner and Tresselt digraph statistics. [b]The performance estimation of the OPTI design changes significantly with the percentage of optimal choice of the Space keys, which requires looking ahead of the current character being tapped. If the user makes optimal choice of the four Space keys less than 50% of the time, OPTI performance drops to 36 wpm. [c]If the user makes optimal choice of the two Space keys less than 75% of the time, Fitaly performance drops to 35.2 wpm. Further, the Space bar is calculated twice as wide as other keys. This is only true to lateral movement. [d]The diameter of the inscribed circle of the hexagon keys was used as the target size. Efficiency drops slightly if square keys are used. For example, at 4.9 bits/sec, the ATOMIK layout (Figure 23) changes from 41.2 to 39.9 wpm in case of square keys.

Figure 10. **The FITALY keyboard.**

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

width of the Space keys was considered twice the size of a regular key in the Fitts' law calculation. This clearly was an overestimate when the movement was primarily vertical. Second, there was again the issue of which Space key to use in calculating distances. Two methods were used to deal with this issue. The first always used the closest Space key to each character, with "free warp-

ing" between the two Space keys. By this method, the FITALY keyboard performance was estimated as 37.07 wpm. The second method used the shortest character–space distance 75% of the time. The rest of the time the farther Space key was used in calculating distance. By this method, performance of 35.2 wpm was found.

In summary, the movement efficiency of the FITALY keyboard is about 36 wpm, far more efficient than Qwerty, as the company advertised, but less efficient than OPTI II.

Chubon

Figure 11 shows the Chubon keyboard layout. Using the same approach as in the case of the Qwerty layout, we estimated its movement efficiency to be 33.3 wpm, assuming free warping. If we assume the user always taps at the center of the Space key, the movement efficiency will be 32 wpm. Both estimates are slower than OPTI and FITALY but still faster than Qwerty.

Lewis–Kennedy–LaLomia

Instead of using various heuristics to generate a layout, Lewis, Kennedy, and LaLomia (1999) used a more systematic method in their design process. They first created a symmetrical matrix of the relative frequency of unordered English-language digraphs and then analyzed this matrix with a "Pathfinder network-definition program" to create a minimally connected network, which formed the basis for their design. Because the method does not consider all digraph connections, one cannot expect a truly optimized layout from such an approach.

The same issue of location and size with regard to the spacebar exists in evaluating Lewis et al's design (Figure 12). We use the midpoint of the spacebar, a conservative approach, in our evaluation. The result is that the Lewis–Kennedy–LaLomia layout speed is 37.14 wpm.

Lewis, LaLomia, and Kennedy (1999) also referenced an earlier design—the modified Getschow, Rosen, and Goodenough-Trepagnier (1986) layout (Figure 13). Our analysis shows that such a design has a speed of 37.8 wpm.

3. DEVELOPING QUANTITATIVE DESIGN METHODS

We have reviewed a few virtual keyboard alternatives to the Qwerty layout. The layout that produces the highest movement efficiency is MacKenzie and Zhang's OPTI II keyboard. Is the OPTI II keyboard the optimal virtual keyboard design? Can we design a virtual keyboard that facilitates higher movement efficiency? Most of the existing designs, at least in part, are based on

Figure 11. **The Chubon keyboard.**

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

Figure 12. **The Lewis–Kennedy–LaLomia layout.**

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

Figure 13. **The revised Getschow–Rosen–Goodenough–Trepagnier Layout by Lewis, LaLomia, and Kennedy (1999).**

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

manual trial-and-error approaches, with the help of letter and digraph frequency tables or a minimally connected digraph network. Given the great number of possibilities, human manual exploration can only try out a small fraction of arrangements.

Getschow et al. (1986) introduced one algorithmic approach to virtual keyboard design. They used a "simple assignment procedure called greedy algorithm" that placed alphabetical letters in the most easily accessible positions according to the letter's frequency rank order (pp. 396–398). As the authors stated, the greedy algorithm ignores many arrangements that could be substantially better because it does not consider the letter placement with respect to each other.

The opposite approach to the simple greedy algorithm is exhaustive algorithmic searching that calculates the efficiency of each and every combination of letter arrangement. However, the complexity of that search—$O(n!)$—is approximately 10^{28}, a number too large even for modern computing.

Between the two extremes, we designed and implemented two systematic, physics-based techniques to search for the optimal virtual keyboard. In this section we present the methodologies and results of these two techniques.

3.1. Dynamic Simulation Method

As shown earlier, the goal of good virtual keyboard design is to minimize the statistical travel distance between characters. The more frequent digraphs should be closer together than less frequent digraphs. To achieve this goal, we first designed a dynamic system technique. Imagine a spring connecting every pair of the 27 keys whose initial positions were randomly placed with spaces between the keys. The elasticity of each spring, when turned on, was proportional to the transitional probability between the two keys so that keys with higher transitional probability would be pulled together with greater force. In addition, there is viscous friction between the circle-shaped keys and between the keys and their environment. The steady state when all keys are pulled together forms a candidate virtual keyboard design. Figure 14 illustrates one part of this dynamic system model.

Fortunately, we did not need to build physical models to create the spring–viscosity–mass dynamic systems. Instead we used a mechanical simulation package (Working Model) to simulate it. In the simulation, the springs were "virtual." They did not stop other objects passing through them, hence preventing the springs from being tangled.

The final positions of the keys might still not be at the minimum tension state because some keys could block others from entering a lower energy state. Two methods were used to reduce the deadlock or local minimum states. First, we experimented with different initial states, which had a very significant impact to the end result. Second, each spring had an extended segment (a strut) that held the keys apart so other keys could be pulled through these gaps to reach a lower level of tension. The length of this segment was manually adjusted in the dynamic simulation process. At the end of each simulation cycle, we reduced the length of the adjustable struts to zero so all the keys were pulled against each other, forming a layout of a virtual keyboard. The movement efficiency of the design was then calculated according to Equation 2 and compared with known results. When unsatisfactory, the layout could be "stretched" out to serve as another initial state for the next iteration of the same process. The iteration was repeated until a satisfactory layout was formed. Figure 15 shows the most efficient layout we achieved with this approach. To capture the gist of the spring simula-

Figure 14. **(Part of) the dynamic simulation model: Frequent digraphs are connected with stronger springs.**
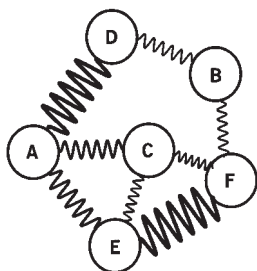
Figure 15. **Hooke's keyboard.**
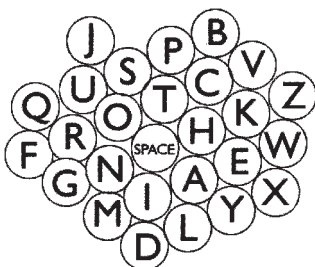
tion technique, we call it Hooke's keyboard (after Hooke's Law). The move-ment efficiency of the Hooke's keyboard shown in Figure 15 is 41.6 wpm, higher than the most efficient previous design (OPTI II, 38 wpm).

3.2. Fitts-Digraph Energy and the Metropolis Method

The idea of minimizing energy, or tension, in the keyboard layout brought us to explore a better known optimization method—the Metropolis algorithm. The Metropolis algorithm is a Monte Carlo method widely used in searching for the minimum energy state in statistical physics (Binder & Heermann, 1988; Metropolis, Rosenbluth, Rosenbluth, Teller, & Teller, 1953; see Beichl & Sullivan, 2000, for a recent review of the Metropolis algorithm). If we define Equation 2 as "Fitts-Digraph energy," the problem of designing a high-perfor-mance keyboard is equivalent to searching for the structure of a molecule (the keyboard) at a stable low energy state determined by the interactions among all the atoms (keys). Applying this approach, we designed and implemented a software system that did a "random walk" in the virtual keyboard design space. In each step of the walk, the algorithm picked a key and moved it in a

random direction by a random amount to reach a new configuration. The level of Fitts' energy in the new configuration, based on Equation 2, was then evaluated. Whether the new configuration was kept as the starting position for the next iteration depended on the following Metropolis function:

$$W(O - N) = e^{-\Delta E / kT} \ \ if \ \ \Delta E > 0$$
$$= 1 \quad\quad if \ \ \Delta E \leq 0 \tag{4}$$

In Equation 4, $W(O - N)$ was the probability of changing from configuration O (old) to configuration N (new), ΔE was the energy change, k was a coefficient, and T was "temperature," which could be interactively adjusted. The use of Equation 4 makes the Metropolis method superior to our previous spring model because the search does not always move toward a local minimum. It occasionally allowed moves with positive energy change to be able to climb out of a local energy minimum.

Again, the initial state where the random walk starts from had a significant impact on the search process. An existing good layout stretched over a larger space was used as an initial state.

In addition to the automatic random walk process itself, we also applied interactive "annealing," as commonly used in the Metropolis searching process. The annealing process involved bringing temperature (T in Equation 4) through several up and down cycles. When temperature was brought up, the system had a higher probability of moving upward in energy and jumping out of local minima. When temperature was brought down, the system descended down to a lower energy level. This annealing process was repeated until no further improvement was seen. Figures 16, 17, and 18 are snapshots from the Metropolis random walk process in one annealing cycle.

We call layouts produced by this process *Metropolis keyboards*. Various layouts with similar movement efficiency were produced. One of them is shown in Figure 19, in which we have replaced the circle shapes used in the design process with hexagons. Each hexagon encapsulated the circle it replaced and filled the gaps between the circles, making more efficient use of the total space. Applying Equation 2, we calculated the movement efficiency of this layout at 42.1 wpm,[4] using the Mayzner and Tresselt table. This is a 50% improvement over Qwerty and more than 10% improvement over OPTI II.

---

4. The diameter of the inscribed circle of a hexagon was used in our Fitts' law calculation. Hence, this is a slightly conservative estimate. If we use the average of diameters of the inscribed and circumscribed circles of a hexagon, the performance here will be 43.7 wpm.

Figure 16. Early stage of Metropolis random walk. The system is at a high energy state, moving toward a lower energy state.
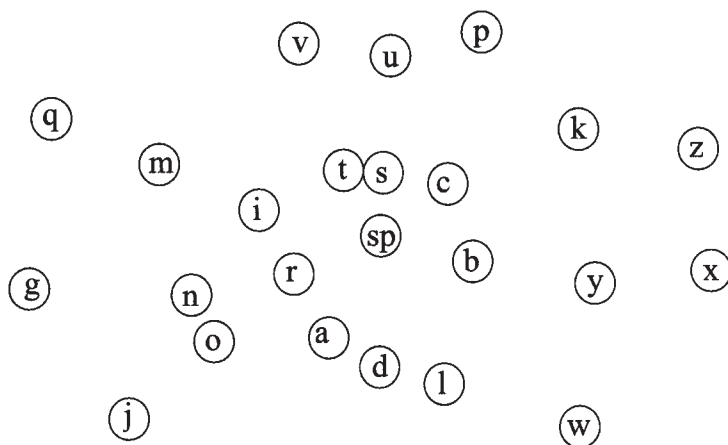
Figure 17. Middle stage of Metropolis random walk. Keys have been descended to a lower energy state. They are getting packed.
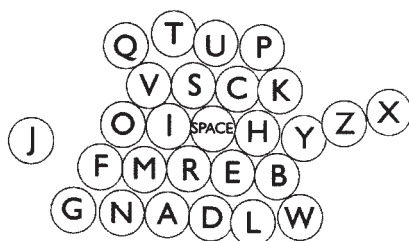
Figure 18. Later stage of Metropolis random walk. Keys moved to a lower still energy state.
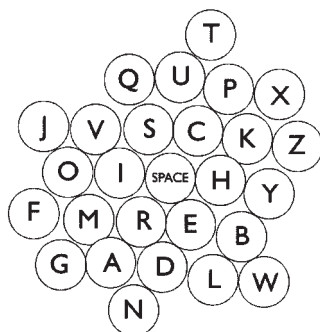
When we recalculated the speed of the layout in Figure 19 by our chat and news digraph table, we found 41.63 wpm performance by the chat digraph table, and 41.37 wpm by the news table, respectively. The similarity of movement efficiency measured by different corpora is somewhat surprising, given the very different language styles the corpora represent. We initially thought we might have to design virtual keyboards specifically for each type of application, but this is clearly not necessary given the small performance differences. One explanation for the insensitivity of the layout to the corpus is that the language style (formal vs. informal) does not significantly affect the phonology of a corpus, which dictates the digraph distributions. In other words, as long as the language sounds English, the digraph distributions should remain similar. A different language with distinct phonology may indeed require a different layout, although the methods presented here still apply.

The layout shown in Figure 19 does not form a rectangle shape. To make it a rectangle, keys on the outside can be rearranged without great impact on the movement efficiency due to their low frequency. For example, if we move the *J* key to the left of *F* key and *B* key to the left of *D* key, the total movement efficiency of the Metropolis keyboard in Figure 19 would only decrease to 41.6 wpm (using the Mayzner and Tresselt table).

Alternatively, instead of performing the random walk algorithm on an open space where each walk step was taken by moving a randomly selected key to a random direction by a random distance, we could and indeed have developed a program that uses the Metropolis method in a confined array of hexagons. Each walk step was taken by swapping a random pair of keys. Whether such a step took hold depended on the Metropolis function (Equation 4). The rest of the random walk process was the same as the previous approach. Our experience shows that this is an equally effective and more efficient approach, which was used to produce the layouts presented in the rest of the article.

3.3. A Variety of Layouts

By means of the Metropolis algorithm, we designed a variety of layouts with different characteristics, all with similar movement efficiency. This means that it is possible to accommodate design considerations other than the Fitts-digraph energy function. Figure 20 is a layout we accidentally discovered. A particularly interesting characteristic of this layout is that the vowels are connected symmetrically, equally dividing the keyboard into three regions and making the layout more structured. The movement efficiency of this layout is 43.1 wpm, based on the Mayzner and Tresselt digraph statistics. When we applied our digraph tables constructed from current news and chat room text corpora, the movement efficiency was 42.2 wpm and 42.3 wpm, re-

Figure 19. **Metropolis Layout 1, with movement efficiency of 42.1, 41.63, and 41.37 wpm based on the Mayzner and Tresselt, chat, and news corpora, respectively.**
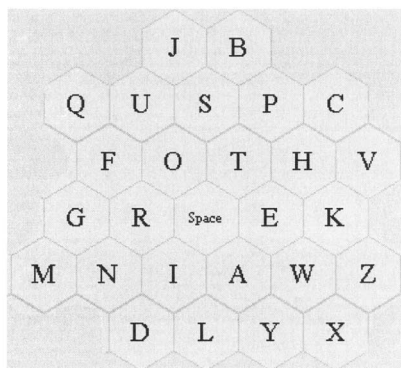
spectively. Whether this more structured layout is beneficial requires future research.

The Metropolis method can also be used to design a virtual keyboard with any shape desired. All we need to do is lay out the keys in that shape and then let the random walk (swapping) process take over the optimization. Figure 21 gives a triangle example, with 42.45 wpm movement efficiency.

4. ALPHABETICAL TUNING AND WORD CONNECTIVITY: THE ATOMIK LAYOUT

4.1. Alphabetical Tuning

Given the flexibility of producing a variety of layouts with the Metropolis method, we decided to introduce additional characteristics to a layout that may benefit users' learning experience. For novice users of virtual keyboards, speed is determined mostly by the needs to search and find target keys rather than by the amount of motor movement. A keyboard optimized by movement efficiency only may look rather arbitrary to a novice user and hence be difficult to search. We explored the possibility of easing the novice user's search process by introducing alphabetical ordering to a virtual keyboard layout. Alphabetical layout is not a new idea. For example, Norman and Fisher studied a strictly alphabetical layout of the physical keyboard of a typewriter (Norman & Fisher, 1982). They expected, but did not find, that novice users typed faster on such a keyboard than on a standard Qwerty keyboard. The main problem with an alphabetical keyboard, they concluded, was that the keys were laid out sequentially in multiple rows. The location of a key depended on the length of

Figure 20. Metropolis Layout 2, with aligned vowels that divide the keyboard to three regions. Its movement efficiency is 43.1, 42.2, and 42.3 wpm based on the Mayzner and Tresselt, chat, and news corpora, respectively.
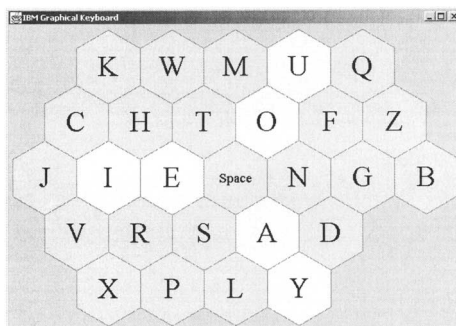
Figure 21. A triangle-shaped layout with 42.45 wpm movement efficiency.
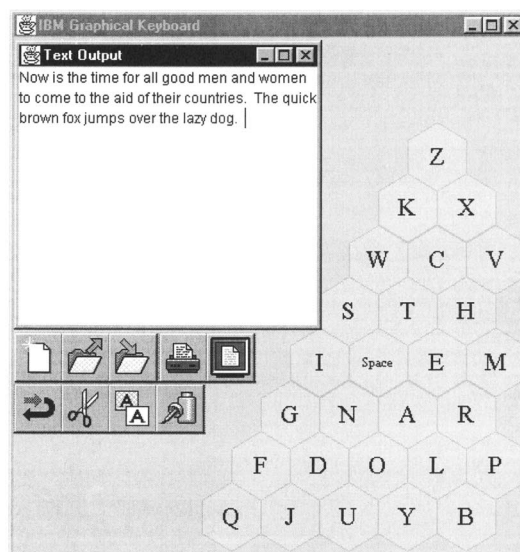
each row—the break point from which the next letter had to start at the left end of the keyboard again. MacKenzie, Zhang, and Soukoreff (1999) studied a virtual keyboard where the letters were laid alphabetically in two columns. Again, they did not find a performance advantage with the alphabetical layout. A confounding factor that might have diminished typing speed in this study was the elongated two-column shape that required an increased average MT. Lewis, LaLomia, et al. (1999) proposed a 5×6 virtual keyboard layout

with a strictly alphabetical sequence (see Figures 5 and 6). Such a design should suffer from the same problem as discovered by Norman and Fisher—the alphabetical discontinuity caused by row breaks.

Instead of strictly laying out the keys in an alphabetical sequence, we introduce *alphabetical tuning* in the optimization process. To produce such a keyboard, an additional term was added to the "energy" function, which, for each key, depended on the place in the alphabet for the character and on its position on the keyboard:

$$(5)$$

$$e = t + \lambda \sum_{i=a}^{z} \eta(i)(y_i - x_i),$$

where t was the previous energy term defined by Equation 2. λ was an empirically adjusted weighting coefficient, depending on how much alphabetical order was brought to consideration at the cost of the average MT. $\eta(i)$ was an integer number representing the place of the letter in the alphabet, with $\eta(a) = -12$, $\eta(b) = -11$, …. $\eta(m) = 0$, $\eta(n) = 1$, … and $\eta(z) = 13$. x_i and y_i were the coordinates of letter with origin $(0, 0)$ at the center of the keyboard. The term $\eta(i)(y_i - x_i)$ can be viewed as two forces. $\eta(i)y_i$ produced a force pushing the first half of the letters (a to m) upward and the second half (n to z) downward, with a resulting energy proportional to letter positions. For example, for letter a, $\eta(a) = -12$. The lowest energy state for it is the uppermost position and the highest energy state lies in the lowermost position (negative). The opposite is true for letter z, $\eta(z) = 13$. Similarly, the other force, $\eta(i)(-x_i)$ pushes the first half the letters (a to m) leftward and the latter half rightward. For the Space key, a special case in this treatment, the alphabetic bias term was zero at the center of the keyboard and increases exponentially with distance from the center.

The result of Equation 5 as an objective function was the general trend of letters starting out from the upper left corner moving toward the lower right corner. Figure 22 shows one example of such a design. Clearly, the general tendency of alphabetical order was preserved by this approach, without a significant sacrifice of movement efficiency (41.8 wpm for chat and 41.7 for news).

We conducted a study to test if alphabetical tuning indeed helps novice users. Here we only report a brief summary of the novice user study and refer the reader to Smith and Zhai (2001) for more details. In the study, 12 users with no prior experience with virtual keyboards participated in an order-balanced within-subject experiment with the two layouts, one with alphabetical tuning (shown in Figure 22) and one without alphabetical tuning. The two layouts shared exactly the same geometry (shape and size). With each layout, participants first tapped from the a to the z key as a brief warmup. For the next 15

Figure 22. An alphabetically tuned layout. The first letters tend to appear in the upper left corner, and the last letters tend to appear in the lower right corner.
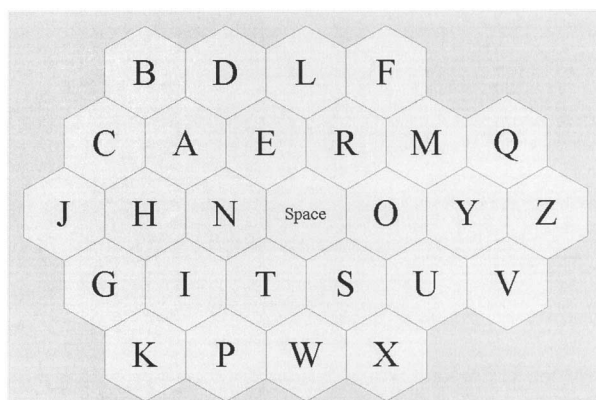
min, they entered memorable English sentences, such as "the quick brown fox jumps over the lazy dog," "we hold these truths to be self-evident," and "all men are created equal." Results show that participants' average speed was 9.7 wpm on the keyboard with alphabetical ordering and 8.9 wpm on the keyboard without alphabetical ordering. The speed difference (9%) between the two conditions was statistically significant, $F(1, 11) = 6.74$, $p < .05$. The error rates were 2% with the alphabetical order and 2.2% without the alphabetical order, $F(1, 11) = .55$, $p = .47$, *ns*.

To explain the empirical findings of alphabetical tuning, Smith and Zhai (2001) also conducted a theoretical analysis of the uncertainty in visual search, in the framework of the Hick–Hyman law (see Keele, 1986, for a review). With the alphabetical tuning, novices may have a stronger expectation of the area that a letter is likely to appear (lower entropy), hence reduce their search time. This is particularly true for the first (a, b, c, d, e...) and last letters (u, v, w, x, y, z).

We call layouts generated with this approach *alphabetically tuned and optimized mobile interface keyboard* (ATOMIK) layouts to reflect both the efficiency and the alphabetical tendency characteristics, as well as the method by which they were produced—atomic interactions between the keys.

4.2. Connectivity of Frequent Words

In using the various layouts produced thus far, we found that the connectivity of a word—the degree to which consecutive letters in the word are adjacent—is very important to movement efficiency, visual search, and memory of the pattern of the word. These motor, visual, and cognitive benefits were not fully characterized by the Fitts-digraph energy. This is particular important to

the most common words such as *the*. If a user frequently types a word that is tightly connected, it may ease the user's effort, both real and perceived, hence enhancing the usability and the initial subjective acceptability of virtual keyboards.

We therefore introduced the third criterion in our design—Connectivity Index (CI). CI was defined as

$$CI = \sum_{i=1}^{N} F(i)c(i), \tag{6}$$

where $f(i)$ is the percentage frequency of the ith most frequent word and $c(i)$ is the connectivity score of that word. For example, for the word *the*, if t–h and h–e are connected (adjacent), the word *the* gets a score of 1: $c(i) = 1$. It is multiplied by $f(i) = 3.38\%$, its frequency, before being added to CI. If only t–h or only h–e are connected, the word *the* gets a score of $c(i) = .5$.

In consideration of the Zipf's law[5] effect, we only used the most frequent words to compute CI. These top-ranked words cover a disproportional amount of usage (Figure 23). Excluding single letter words *I* and *a*, we chose the top 17 most frequent words in the chat corpus to compute CI. They were *the, to, you, and, of, is, that, in, it, no, me, are, with, have, was, for,* and *what*.

Figure 24 shows an ATOMIK layout that well satisfies all three criteria—movement efficiency, alphabetical tuning, and word connectivity. The movement efficiency is 41.67 wpm, based on the Mayzner and Tresselt digraph statistics (Mayzner & Tresselt, 1965); 41.16 wpm, based on the chat digraph statistics (Appendix Figure A–1); and 40.78 wpm, based on the news digraph statistics (Appendix Figure A–2).

Because many of the common words are totally connected (e.g., *the, to, and, is,* and *in*), experienced users might be able to stroke through these letters instead of tapping on each of them. Such a strategy may not only save time but also enrich the set of input gestures.

5. ADDITONAL OPTIMIZATION ISSUES

5.1. Auxiliary Keys

Auxiliary keys include all nonalphabetic keys, such as the punctuation keys. How should these auxiliary keys be arranged? Should we optimize them

---

5. Zipf's law models the observation that frequency of occurrence of some event f, as a function of its rank i, defined by the frequency, is a power-law function $P_i \sim 1/i^a$ with the exponent a close to unity. Figure 23 shows the word percentage distribution as a function of its frequency rank in our chat corpus.

Figure 23. **Word frequency distribution over rank in our chat corpus.**
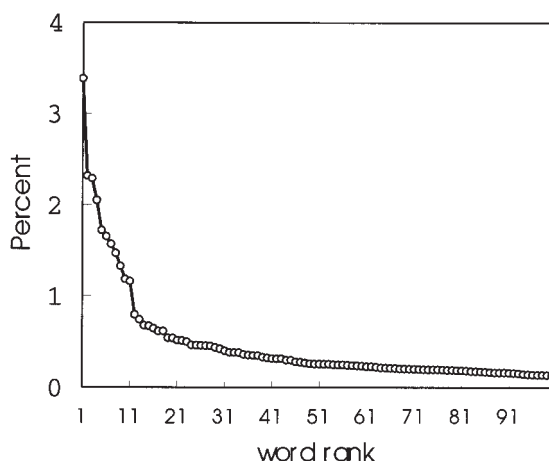
Figure 24. **An ATOMIK layout with high degree of efficiency, alphabetical tuning, and word connectivity. The movement efficiency is 41.67 wpm, based on the Mayzner and Tresselt digraph statistics; 41.16 wpm, based on the chat digraph statistics; and 40.78 wpm, based on the news digraph statistics. (Copyright IBM).**
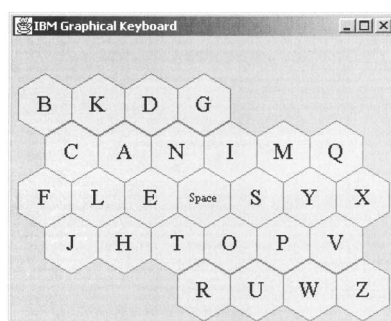
together with the letter keys? Many of the punctuation keys do in fact have higher frequency than some letter keys. For example, the frequency of period (.) is higher than many other alphabetic letters (*v, k, j, x, z,* etc.; Figures 3 and 4). From the ultimate performance perspective, at least the more frequent punctuation keys should be optimized with the letters key simultaneously.

On the other hand, mixing the punctuation keys with the letter keys will reduce the structural order of the keyboard and increase the search space for the novice users. If all the letter keys are arranged together, the user has to search 26 positions at the most to look for any particular letter key. Otherwise, the user has to search among a greater number of positions, making the keyboard

more difficult to learn. It is conceivable that if only a few most frequent keys (e.g., Period and Shift) are mixed with letter keys, the distinct appearance of these keys from the letter keys may become "landmarks" among the letter keys. It has been shown that the presence of visually silent landmarks helps the users to remember the location of the graphical objects around these landmarks (Ark, Dryer, Selker, & Zhai, 1998). However, the cost of mixing a few auxiliary keys with letter keys is the inconsistent organization of keyboard. If all letter keys are together at the core of the keyboard and all auxiliary keys are on the outside, such as in the examples shown in Figures 25 and 26, the more consistent separation may help the novice user to learn about the keyboard. To enhance visual structure, auxiliary keys were given a very different appearance (Figure 26).

According to our digraph table (Appendix Figure A–2), the numeric characters are more frequently connected with each other (and space) than with alphabetic characters. This argues for placing the numeric keys together in a number pad, such as the telephone pad arrangement shown in Figure 25.

Note that the auxiliary keys increase the total number of keys on a keyboard and hence increase the average movement distance, which in turn decreases the total performance of the keyboard. To hold a consistent comparison standard, we always use the letter and Space keys only in the calculation of wpm.

To save space, two auxiliary keys can share one location multiplexed by a Shift key. Figure 25 shows an ATOMIK keyboard implementation with the multiplexing feature. Space saving is particularly necessary on small screens. Figure 26 shows an ATOMIK keyboard implementation on a handheld PDA. Due to the limited screen resolution of the PDA, which causes steps in the rendering of the diagonal edges of hexagons, square-shaped keys were used in this implementation.

5.2. Multiple Space Keys and Varying Key Sizes

Both the OPTI keyboard and the FITALY keyboard used more than one Space key to accommodate the high frequency of space in English writing (Figures 8 and 10). We decided against such an idea for the following reasons. First, multiple Space keys take more space from the real estate available to regular keys, which may reduce the total efficiency of the keyboard. Second, as revealed by our analysis, the performance of a multiple Space key keyboard highly depends on the user's optimal choice of the Space key, which requires planning one key ahead of tapping. Third, the Space key is not the only one with high frequency (Figures 3 and 4).

Related to the multiple Space key issue is the size of the Space key. Should the Space key be given a greater size than other keys? Should other more frequent keys also be given greater size? According to Fitts' law, tapping time is

Figure 25. An ATOMIK keyboard implementation with auxiliary keys (Copyright IBM). Shift key press alternates the functions of the auxiliary keys as well as the letter cases from (a) to (b).
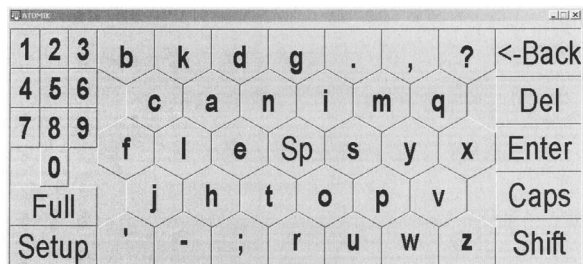
(a)
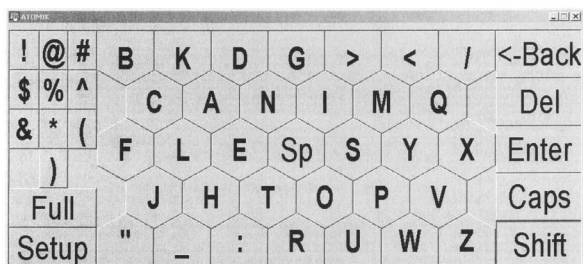
(b)

Figure 26. Two ATOMIK keyboard implementations (Copyright IBM) on a handheld PDA. Some of the keys are multiplexed by the shift or cap key press. The keys are made into square shape in this implementation, which slightly reduces movement efficiency to 39.9 wpm, based on Mayzner and Tresselt digraph statistics.

257
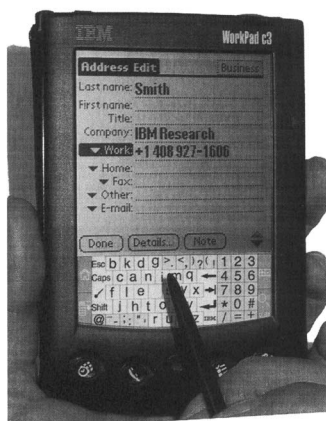
related to both distance and target size. We have optimized the statistical distance to reduce tapping time. By the same principle, shouldn't we also optimize the relative key sizes so more frequent keys are given a greater share of the real estate?

We have indeed explored the issue of varying key sizes, but we have not come to a positive conclusion. There are at least four issues to resolve. First, the optimization of both size and distance is much more complex. One of the complicating factors is that keys of unequal size cannot be as tightly packed and still be optimized in positional layout. We have made several versions of search algorithms to optimize both the key sizes and the position layout, but so far we have not produced a keyboard that had higher movement efficiency than the Metropolis keyboards with a constant key size.

Second, from Fitts' law point of view, frequently used keys should be given a greater size. However, these frequent keys should also be placed toward the center of the keyboard. Crossing these bigger keys to reach other keys introduces a performance penalty.

Third, there is an asymmetrical effect to size gain and loss in Fitts' law. To reciprocally tap on the two adjacent targets shown in Figure 27, the performance gain of tapping the enlarged right target is less than the loss of tapping the reduced left target. Figure 28 illustrates tapping time from the left to right and vice versa. As we can see, as the asymmetry factor x changes from 0 to positive, the time reduction from the left to the right target does not compensate for the time increase in the reverse direction. The lowest sum of the two is when $x = 0$, assuming equal frequency of entering the left and the right target.

Fourth, even if we found a more efficient layout with varying key sizes, there could be a cost of varying control precision depending on which letter is being typed. The loss of consistency in control precision may be detrimental.

In summary, varying size remains an open problem, although the factors we have considered suggest against it.

5.3. Upper Bounds of Virtual Keyboard Optimization

Has our exploration achieved the maximum efficiency? We have run a large number of simulations, but all reached a similar plateau of movement efficiency, suggesting we are very close, if not at, the maximum efficiency. Figure 29 shows one trace of a Metropolis random walk optimization trial in terms of wpm speed. However, it is theoretically interesting to estimate the lowest upper bound movement efficiency. To that end, we have produced three reference points, based on three physically impossible layout designs. The first was that all keys were co-located in one spot, hence the user only needed to tap on the same spot. With such a hypothetical keyboard, the estimated performance was 95 wpm. The second estimate assumed the next key needed was always next to

Figure 27. **Asymmetrical Fitts' targets, with asymmetry coefficient factor *x*.**
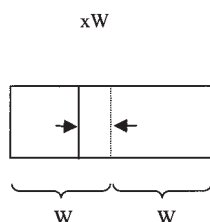
Figure 28. **Tapping time from left to right key, vice versa, and the sum of the two as a function of the asymmetry coefficient *x*. The lowest sum occurs when *x* = 0; that is, the two targets are equal in size.**
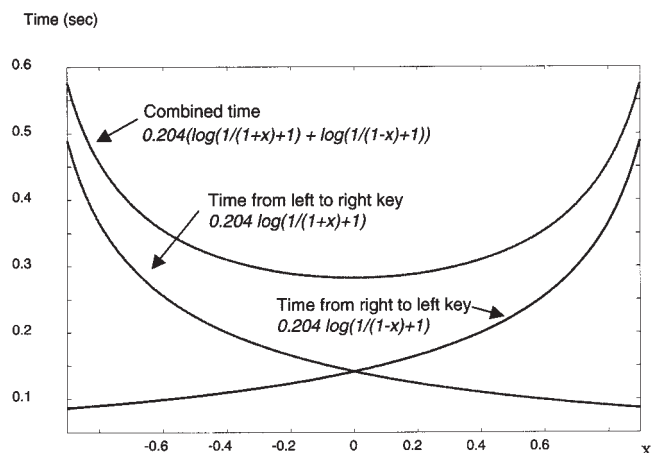
the current key, requiring tapping with Fitts' ID = 1 bit. Performance dropped to 59 wpm with this assumption (at 4.9 bps Fitts' IP). In the third estimate, when calculating Fitts' law performance for any key, the rest of the keys were optimally placed according to their digraph frequency to the current key. The performance of this more realistic but still impossible keyboard was 53 wpm, about 10 wpm faster than our most efficient design. Finding the lowest upper bound of virtual keyboard performance is another open research issue.

6. DISCUSSIONS

6.1. User Interface Design Techniques

Although sharing the same ultimate goal, user interface design and user interface evaluation are traditionally two entirely separate processes involving

Figure 29. **A Metropolis random walk trace in terms of wpm speed.**
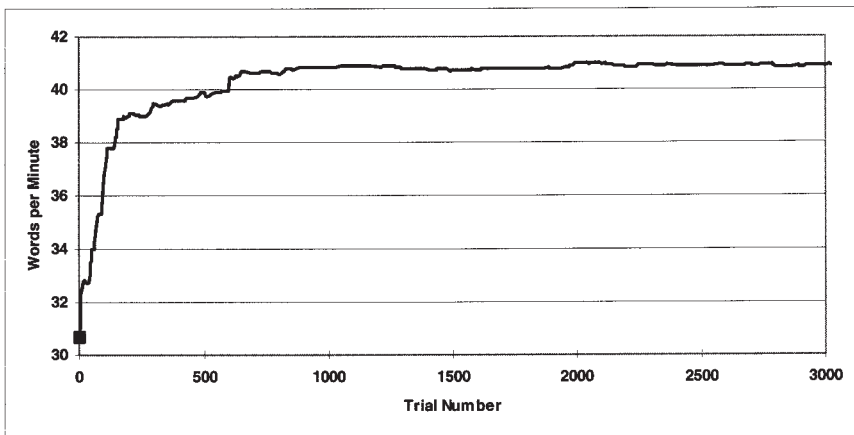
different methodologies. User interface evaluation tends to be measurement based, whereas user interface design tends to be intuition, heuristics, and experience based. The design exploration presented here is a departure from that norm. First, the design process was quantitative and computerized. Second, the design process was integrated to the highest degree with evaluation—every step of the design space search was guided by the evaluation function. Third, the quantitative design process was based on previous evaluation research, particularly the work on Fitts' law. Without Fitts' law, we could still construct an evaluation function simply based on the digraph frequency and travel distance, so we could know the relative superiority of one layout to another based on statistical distance.[6] We would not, however, be able to relate the statistical distance to user performance. With Fitts' law modeled in the evaluation function, we could instantly estimate the eventual average user performance and compare it to known benchmarks.

The optimization methods and software presented here can be used to optimize keyboard layout against any quantifiable objective function and any language corpus. As shown in the design of the ATOMIK layout, we could combine multiple objectives in the same optimization process. However, the relative weights of the difference objectives are design choices that require careful research. In the case of the ATOMIK design, we chose efficiency as the

---

6. Note, however, a layout based on distance optimization may be different from, and less valid than, a layout based on MT optimization due to the nonlinear relation between distance and MT in Fitts' law.

predominant, alphabetical ordering as the secondary, and connectivity index as the third objective.

6.2. Limitations of This Work and Future Research Issues

This study has focused on the optimization of virtual keyboards. We use the term *optimization* in the mathematical sense—searching for the lowest (or highest) value of an objective function with given constraints. The basic objective function we have used is the Fitts-digraph energy defined by Equation 2, which quantifies movement efficiency when using the virtual keyboard with a single stylus. Later, we added alphabetical ordering and word connectivity as additional objectives. The resulting layouts of this optimization process are by no means the "best" in the general sense because there could be many other factors influencing the usability of a virtual keyboard.

Furthermore, movement efficiency in this study was based only on digraphs, without considering how consecutive tapping movements affect each other. It is conceivable that the angle between two consecutive tapping movements has an impact on performance.

We should also reemphasize the assumptions made to calculate the estimated movement efficiency numbers in wpm, which are primarily to serve as indexes for performance comparison. These estimated numbers are not necessarily every user's actual speed at any time. For example, the 41.67 wpm movement efficiency of the ATOMIK layout (Figure 24) was based on the following assumptions:

1. The text entered follows the same digraph distribution of the Mayzner and Tresselt corpus, although we have shown that the Fitts-digraph energy was not very sensitive to the style of text.
2. Every five key strokes, including the Space key, counts for one word.
3. The user tapping performance, as measured by Fitts' law, is at 4.9 bps. If not, the estimated number scales proportionally.
4. The diameter of the inscribed circle of the hexagon keys was taken as the width of target in Fitts' law calculation.
5. The time taken in visual search or other cognitive components in text entering is negligible in comparison to the stylus MT, which is definitely not true for novice users.

A UI design is not complete without user studies. A user study merely to confirm the movement efficiency predictions presented here, however, would not be very informative. First, the very foundation of our calculation, Fitts' law, has already been repeatedly tested. Second, previous user studies have shown that users could indeed master a new layout and eventually

reach the performance level of over 40 wpm on the OPTI design (MacKenzie & Zhang, 1999), after twenty 45-min practice sessions.[7] Would an average user be willing to invest 20 hr for 50% performance gain? Twenty hours is probably an unacceptably long period for learning a UI technique, although it is only a fraction of a typical physical Qwerty typist's learning time. We are currently researching training methods that can accelerate the learning curve. Many deeper research questions have to be answered before an effective training method can be found. How do users learn the keyboard layout? Do they learn the paths of words or do they learn the positions of the keys? Our observation to date suggests a combination of both, initially with letter positions and later shifts toward higher level patterns. There is a body of literature on learning typing on a physical keyboard (e.g., Cooper, 1983; Dvorak et al., 1936; Ono & Yamada, 1990), but there is no reason to believe the mechanism involved in tapping on virtual keyboard is the same as 10-finger typing on a physical keyboard. Another class of questions about learning is whether one layout with a particular property is faster to learn than another. We have tried adding landmarks or dividing the keyboard to regions by color coding, but no obvious learning advantage has been gained. There are many fascinating cognitive issues to be investigated in the future.

7. CONCLUSIONS

Motivated by the increasing importance of pervasive computing devices and built on the previous work on virtual keyboards (e.g., Getschow et al., 1986, MacKenzie & Zhang, 1999; Soukoreff & MacKenzie, 1995), this article explored the design of optimized virtual keyboards and made the following contributions. First, the article thoroughly analyzed the movement efficiency of existing virtual keyboards and corrected erroneous estimations of virtual keyboards in the literature. We found the movement efficiency of Qwerty, FITALY, and OPTI keyboards to be in the neighborhood of 28, 36, and 38 wpm, respectively (see Figure 9 for a complete summary). Second, we introduced two quantitative techniques to virtual keyboard design. One technique used physical simulation of digraph springs, producing a Hooke keyboard with 41.6 wpm movement efficiency. The other method used the Metropolis

---

7. A casual user of the layout in Figure 19 "got up to speed" after "about 20 hr of use over 5 days." He wrote a 10-page report and estimated that his speed was "at least twice as fast as Graffiti" (personal e-mail communication, 2001). Although this is totally uncontrolled self-reporting, it corroborates with the learning time reported in MacKenzie and Zhang (1999). We expect the learning speed of the ATOMIK layout incorporating alphabetical tuning and word connectivity to be faster.

random walk algorithm, guided by the Fitts-digraph energy object function. This method produced various Metropolis keyboards around 42.5 wpm movement efficiency, which was more than 50% faster than the Qwerty keyboard. The 42.5 wpm was based on a very conservative assumption of 4.9 bps Fitts' law IP and can be scaled up with the IP value. Third, our design exploration led to a variety of layouts with similar speed performance but different design considerations such as the structure and shape of the overall keyboard. Fourth, in addition to movement efficiency quantified by Fitts-digraph energy, we introduced the concept of alphabetical tuning and produced the ATOMIK layouts, which were demonstrated to be easier for novice users. Fifth, we introduced the concept of word connectivity in the final layout, which may further enhance the usability and acceptability of virtual keyboards. Finally, we illustrated the benefits of quantitative design combined with human performance models over traditional UI design methods based on manual trial and error and heuristics. We demonstrated the importance of quantitative techniques and basic human performance modeling to the field of user interface research.

NOTES

Authors' Present Addresses. Shumin Zhai, Department NWE–B2, IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120. E-mail: **zhai@almaden.ibm.com**. Michael Hunter. E-mail: **michael@hunter.org**. Barton A. Smith, Department NEW–B2, IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120. E-mail: **basmith@almaden.ibm.com**.

REFERENCES

Ark, W., Dryer, D. C, Selker, T., & Zhai. S. (1998, March), Landmarks to aid navigation in a graphical user interface. *Proceedings of Workshop on Personalized and Social Navigation in Information Space*. Stockholm, Sweden: ICS.

Beichl, I., & Sullivan, F. (2000). The Metropolis algorithm. *Computing in Science & Engineering, 2*(1), 65–69.

Binder, K., & Heermann, D. W. (1988). *Monte Carlo simulation in statistical physics.* New York: Springer-Verlag.

Cooper, W. E. (Ed.). (1983). *Cognitive aspects of skilled typewriting.* New York: Springer-Verlag.

Dvorak, A., Merrick, N. L., Dealey, W. L., & Ford, G. C. (1936). *Typewriting behavior.* New York: American Book Company.

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.

Getschow, C. O., Rosen, M. J., & Goodenough-Trepagnier, C. (1986). A systematic approach to design a minimum distance alphabetical keyboard. *Proceedings of RESNA (Rehabilitation Engineering Society of North America) 9th Annual Conference.* Arlington, VA: RESNA.

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

Karat, C.-M., Halverson, C., Horn, D., & Karat, J. (1999). Patterns of entry and correction in large vocabulary continuous speech recognition systems. *Proceedings of the CHI 99 Conference on Human Factors in Computing Systems.* New York: ACM.

Keele, S. W. (1986). Motor control. In K. R. Boff, L. Kaufman, & J. P. Thomas (Eds.), *Handbook of perception and human performance* (pp. 30.1–30.60). New York: Wiley.

Lewis, J. R., Kennedy, P. J., & LaLomia, M. J. (1999). 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: HFES.

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

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., 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., & Soukoreff, R. W. (2002). Text entry for mobile computing: Models and methods, theory and practice. *Human–Computer Interaction, 17,* 147–198.

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

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

McClard, A., & Somers, P. (2000, April). Unleashed: Web tablet integration into the home. *Proceedings of the CHI 2000 Conference on Human Factors in Computing Systems.* New York: ACM.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equation of state calculations by fast computing machines. *The Journal of Chemical Physics, 21,* 1087–1092.

Norman, D. A., & Fisher, D. (1982). Why alphabetic keyboards are not easy to use: Keyboard layout doesn't much matter. *Human Factors, 24*, 509–519.

Ono, Y., & Yamada, H. (1990). A cognitive type training model whose speed advancement is derived from those of component tasks. *Behavioral Science, 35*, 238–268.

Smith, B. A., & Zhai, S. (2001). Optimised virtual keyboards with and without alphabetical ordering: A novice user study. *Proceedings of Interact 2001—IFIP International Conference on Human–Computer Interaction.* New York: ACM.

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

Textware Solutions. (1998). *The Fitaly one-finger keyboard.* Retrieved from http://fitaly.com/fitaly/fitaly.htm

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

Yamada, H. (1980). A historical study of typewriters and typing methods: From the position of planning Japanese parallels. *Journal of Information Processing, 2,* 175–202.

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.

Zhai, S., & Smith, B. A. (2001). Alphabetically biased virtual keyboards are easier to use: Layout does matter. *Proceedings of the CHI 2001 Conference on Human Factors in Computing Systems.* New York: ACM.

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

APPENDIX

Digraph Tables Compiled From Text Available on the Internet

Figure A–1. **Digraph table of the 26-letter alphabet and the 10 most frequent nonalphabet letters in the chat room corpus.**

	space	!	"	'	,	-	.	1	:	?	a	b	c	d	e	f	g	h
space	4009	65	1894	215	107	648	675	1359	952	185	21745	9578	9261	7650	3973	6532	6755	10090
!	487	1377	263	4	0	0	7	8	3	65	0	29	0	12	0	0	1	1
"	884	0	2	1	31	1	259	2	0	28	150	78	62	81	40	53	78	141
'	233	0	1	2	7	1	22	0	0	12	41	21	21	217	21	13	13	9
,	9888	0	16	3	21	12	9	384	0	1	9	3	3	2	0	1	0	1
-	882	0	1	7	10	526	18	95	17	1	51	60	92	65	35	42	42	34
.	5105	5	777	12	14	4	13069	19	41	47	304	209	153	97	75	54	89	237
1	708	13	148	1	145	34	40	141	175	3	10	2	9	3	3	1	4	55
:	2331	2	7	0	0	108	0	221	256	1	4	1	2	17	0	0	1	2
?	1019	105	325	5	2	0	52	0	0	1845	1	0	0	0	0	4	0	2
a	7498	86	27	53	259	79	304	6	128	112	465	1809	2574	2828	95	402	1543	1653
b	245	16	8	6	33	10	50	0	14	16	1810	290	34	115	5033	8	9	10
c	777	8	6	18	97	12	199	0	53	49	4074	15	446	46	3020	2	9	5429
d	14705	114	94	141	696	58	1065	0	160	262	1610	34	108	305	4356	48	213	51
e	34370	375	140	514	1271	175	2137	44	272	761	6619	426	2027	5349	3716	990	620	1038
f	5784	20	9	10	73	52	156	3	42	52	1597	0	6	20	1523	1122	89	2
g	5761	57	27	23	249	24	432	2	32	117	1275	12	5	60	2665	13	294	1875
h	4624	101	29	29	526	45	549	9	60	152	12788	52	33	37	21947	22	9	414
i	6295	32	15	1027	99	64	123	0	43	50	1759	687	4239	2615	3136	2381	2018	35
j	51	3	0	2	4	4	13	0	19	0	486	3	11	2	1098	0	5	7
k	2491	72	13	9	178	19	241	8	33	91	701	17	6	11	3063	15	14	381
l	5774	90	38	40	488	95	585	18	113	130	3227	230	165	2253	6082	542	144	17
m	4209	47	20	24	358	24	549	14	71	148	4265	412	74	60	7113	28	13	16
n	12907	146	57	2027	726	149	927	6	169	272	3056	77	1760	7938	5569	265	7641	43
o	10884	133	12	49	394	72	571	13	63	171	282	794	1164	3161	863	4755	630	983
p	1205	33	12	5	95	22	189	7	93	49	1441	12	20	6	3161	12	8	708
q	10	0	0	0	3	2	2	0	2	0	8	2	0	0	1	0	0	0
r	8898	85	51	77	444	68	603	24	105	135	3884	310	750	1500	13277	210	727	100
s	22762	204	137	101	1381	68	1820	23	221	497	3168	111	934	57	6075	58	647	2816
t	23464	159	96	1072	908	110	1415	7	153	592	3453	60	457	45	6273	48	61	25793
u	4320	37	14	332	126	39	165	0	64	84	686	391	1247	713	1016	411	1001	174
v	310	7	0	2	3	2	20	1	2	4	474	6	7	2	7300	1	5	5
w	2298	33	14	25	155	9	304	0	5	71	4759	94	3	21	3235	18	7	4332
x	251	5	5	4	31	21	39	0	11	34	234	0	114	6	125	1	2	5
y	11413	149	52	174	856	66	865	17	173	325	514	272	68	25	1784	101	118	35
z	191	15	0	4	49	13	30	0	58	14	94	0	0	3	227	0	1	2

i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
17781	2193	2323	5576	9862	5798	10647	6528	473	4274	12637	28897	2468	1107	14233	116	6056	75
2	0	4	17	0	3	1	10	1	7	9	1	0	0	2	0	1	0
267	23	18	31	81	108	86	56	4	48	153	168	35	14	214	5	105	1
14	0	2	239	656	29	12	10	0	375	1872	1915	8	212	8	0	6	0
7	0	1	2	22	0	1	5	0	3	0	9	11	2	4	1	2	0
24	43	19	32	70	31	51	47	0	37	82	60	24	20	46	2	45	4
513	43	18	66	155	144	177	52	3	58	175	308	66	21	222	8	173	3
1	6	2	0	2	2	13	1	0	2	19	2	56	0	8	0	0	0
5	1	0	2	0	2	16	186	0	0	4	6	0	0	3	0	3	0
1	0	0	0	12	1	0	0	0	3	2	0	0	1	2	0	0	0
2545	100	1584	7926	2721	17032	71	1150	7	8083	6395	12701	1425	2409	539	87	2854	399
1338	145	4	1831	7	30	2337	6	0	895	257	99	1994	71	24	0	696	10
1316	3	2494	829	8	12	3576	13	6	1083	481	1771	688	3	0	0	186	26
2814	14	3	228	130	520	4263	6	5	749	1038	27	758	50	27	1	582	15
1683	49	342	5488	2220	7563	1049	958	96	13449	8729	3606	181	2544	1036	1313	2232	106
1546	3	15	405	15	9	2872	1	0	1444	64	375	1046	0	2	9	70	0
2153	4	4	546	59	292	3949	0	1	1184	567	43	1108	4	8	0	181	1
7138	6	26	57	234	342	4669	12	3	1041	117	1359	1137	13	48	8	775	0
410	16	1141	3110	2719	16266	3034	633	93	2734	9766	9615	86	1543	27	106	42	203
38	16	8	0	0	5	596	3	0	18	9	11	1072	169	18	0	0	0
2009	169	21	86	13	1171	300	10	0	80	944	63	72	5	50	0	203	0
5395	1	543	7169	143	27	4721	214	0	150	1510	429	716	182	175	6	3350	92
1812	1	10	61	1022	219	2506	876	0	31	1145	54	1037	0	8	1	1208	1
2349	97	1359	616	75	1453	6352	51	19	108	2630	5478	448	180	37	28	2539	21
974	95	1122	3200	4208	11489	4163	1896	6	8040	2002	4866	11567	1130	3698	115	327	80
1724	2	13	2029	46	2	1902	935	0	1911	567	598	581	1	0	1	153	0
0	0	0	1	1	1	1	0	0	0	1	0	768	1	2	0	0	0
4628	11	646	1229	511	909	4490	234	1	901	2702	1991	1387	267	133	10	1974	9
2744	11	364	595	634	616	3901	1177	30	52	2298	8368	2183	17	312	5	356	10
6133	13	12	677	107	92	8430	87	2	2334	2391	1605	1493	49	326	7	1268	38
591	19	197	2667	946	2631	137	1182	2	3885	5087	4058	98	11	10	32	492	69
1583	0	0	8	3	9	489	0	0	7	16	8	7	2	0	0	10	0
3350	5	9	138	7	570	2410	5	0	382	365	34	73	1	242	0	19	1
330	0	1	6	0	0	17	250	0	2	7	197	42	0	3	88	91	0
406	1	82	66	147	61	6542	127	1	82	935	401	182	2	130	1	131	19
124	0	34	6	2	5	55	0	0	1	2	3	27	0	1	0	96	132

Figure A–2. **Digraph table of the 26-letter alphabet and the 10 most frequent nonalphabet letters in the news corpus.**

	space	"	'	'	-	.	0	1	2	'	a	b	c	d	e	f	g	h
space	20	62	3	2	56	19	2	149	58	38	2069	1034	1104	539	454	817	385	762
"	88	0	0	0	0	0	0	0	0	0	5	6	6	1	1	1	3	1
'	101	0	85	0	0	0	0	1	0	1	0	0	0	3	0	0	0	0
,	1103	45	47	0	0	0	11	2	1	0	0	0	0	0	0	0	0	0
-	63	0	0	0	43	0	2	13	7	0	7	15	8	10	4	2	10	3
.	1050	31	33	13	1	21	2	7	7	0	1	1	9	0	0	1	1	3
0	69	0	6	7	8	11	65	2	1	0	0	0	0	0	0	0	0	0
1	25	0	0	13	5	8	23	6	8	0	0	0	0	0	0	0	0	0
2	23	0	0	7	5	8	20	4	6	0	0	2	0	0	0	0	0	0
'	0	0	0	0	0	0	0	0	0	81	5	3	1	1	0	2	1	4
a	622	0	18	31	2	15	0	0	0	0	1	126	288	369	6	78	171	12
b	24	0	1	6	0	7	0	0	2	0	185	21	1	2	417	0	0	0
c	76	1	1	2	0	16	0	0	0	0	402	1	63	1	500	0	0	410
d	1917	0	8	99	13	119	0	0	0	0	208	1	2	25	619	3	12	3
e	3354	2	27	171	18	129	0	0	0	0	606	57	347	954	295	179	87	19
f	629	0	0	6	17	11	0	0	0	0	106	1	0	0	194	156	2	0
g	577	1	2	30	1	25	0	0	0	0	186	0	0	1	260	0	27	198
h	516	1	10	20	9	21	0	0	0	0	658	7	0	10	2039	0	15	0
i	114	0	11	10	1	12	0	0	0	0	194	55	518	387	239	115	231	0
j	0	0	0	0	0	3	0	0	0	0	19	0	0	0	29	0	0	0
k	170	0	2	24	2	19	0	0	0	0	68	1	0	3	272	0	0	3
l	569	1	10	47	13	24	0	0	0	0	496	6	2	231	610	33	2	1
m	220	0	7	18	3	28	0	0	0	0	495	79	4	2	643	3	0	2
n	1579	2	64	112	15	89	0	0	0	0	348	4	231	896	578	53	803	3
o	758	0	5	33	13	18	0	0	0	0	71	53	163	141	18	641	33	41
p	121	0	1	13	1	31	0	0	0	0	240	0	1	1	355	0	0	55
q	4	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0
r	990	1	14	94	23	123	0	0	0	0	444	16	139	116	1401	22	97	3
s	2263	0	27	205	4	252	0	0	0	0	354	4	97	48	683	3	1	354
t	1780	0	38	94	22	85	0	0	0	0	376	3	34	2	952	3	1	2353
u	39	0	8	5	0	35	0	0	0	0	78	84	105	53	112	16	68	0
v	16	0	1	2	0	6	0	0	0	0	58	0	0	0	554	0	0	0
w	147	0	3	6	3	22	0	0	0	0	318	1	1	7	282	1	0	217
x	28	0	1	2	1	2	0	0	0	0	16	0	11	0	10	0	1	1
y	912	0	26	85	2	73	0	0	1	0	44	5	7	3	111	0	1	2
z	7	0	1	4	0	5	0	0	0	0	35	0	0	0	48	0	1	1

i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
1220	124	90	527	804	396	1170	944	30	636	1468	2810	243	109	1118	0	153	6
15	0	0	3	3	3	4	4	0	2	2	14	0	0	14	0	2	0
0	0	0	4	2	4	0	0	0	8	214	30	0	6	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	1	7	9	2	18	14	2	10	16	13	4	2	3	0	8	0
9	1	1	4	8	9	3	11	0	1	19	0	0	1	1	0	1	0
0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	2	3	0	1	2	0	1	6	24	0	0	6	0	3	0
424	19	136	727	281	1433	2	123	10	760	767	1114	99	146	51	6	317	13
114	1	0	182	13	0	160	0	0	138	17	5	188	1	0	0	159	0
183	0	172	149	1	0	697	0	3	111	12	244	83	0	0	0	29	1
245	0	0	24	28	25	145	0	1	77	83	3	85	25	10	0	33	0
110	13	34	386	219	1076	58	140	26	1553	989	266	27	177	119	129	96	12
255	0	0	67	0	0	421	0	0	131	5	97	51	0	0	0	4	0
120	0	3	36	1	56	131	0	0	166	29	23	79	0	0	0	11	0
512	0	0	12	6	36	323	0	0	60	16	131	45	0	7	0	16	0
2	3	45	357	186	2025	474	47	1	259	737	803	12	175	1	18	0	68
20	0	0	0	0	0	82	0	0	0	0	0	46	0	0	0	0	0
101	0	0	7	1	20	35	3	0	9	62	2	0	0	0	0	1	0
508	0	15	419	15	2	269	20	0	6	151	44	71	11	5	0	274	1
269	4	0	1	78	3	271	174	0	66	41	0	54	0	1	0	39	0
302	14	72	49	53	71	282	14	4	5	348	833	70	64	1	0	95	12
99	7	65	199	452	1296	169	207	0	1019	219	344	661	171	264	3	20	4
69	0	1	206	6	0	343	108	0	382	47	49	119	0	0	0	6	0
0	0	0	0	0	0	0	0	0	0	0	0	80	0	0	0	0	0
553	1	121	88	105	152	576	39	1	72	375	248	104	47	5	0	127	0
425	3	36	46	71	14	242	132	9	1	324	850	238	0	13	0	40	1
809	0	0	65	13	10	908	10	0	247	339	138	158	1	56	0	93	6
66	1	5	251	96	344	2	118	1	343	335	355	0	5	0	4	6	0
237	0	0	2	0	0	63	0	0	0	0	1	0	0	0	0	1	0
318	0	0	16	4	70	205	2	0	18	41	9	0	0	10	0	4	0
8	0	0	0	0	0	1	46	0	0	0	30	2	0	0	0	0	0
29	0	3	8	13	9	117	7	0	0	68	5	2	0	0	0	0	0
15	0	0	0	0	0	2	0	0	0	0	0	3	0	0	0	2	2