Entering Text with A Four-Button Device

Kumiko Tanaka-Ishii and **Yusuke Inutsuka** and **Masato Takeichi**
The University of Tokyo
7-3-1 Bunkyoku Hongo, Japan
{kumiko, inu, takeichi}@ipl.t.u-tokyo.ac.jp

Abstract

This paper presents the design of a text-entry device that requires only four buttons. Such a device is applicable as the text interface of portable machines and as an interface for disabled people. The text-entry system is predictive; the basis for this is an adaptive language model. Our evaluation showed that the system is at least as efficient for the entry of free text as the text-entry systems of current-generation mobile phones. The system requires fewer keystrokes than a full keyboard. After adaptation, one user reached a maximum speed of 23 wpm.

1 Introduction

Electronic machinery is becoming smaller; recent developments in palmtop and mobile-phone technologies offer dramatic examples of this process. Since smaller machines are more portable, their users have freer access to information. Here, however, the user interface is a major issue.

If a machine is being used as a medium for person-to-person communications, a natural interface might be speech-based. For other tasks, however, like browsing through Internet pages or looking up databases, the most natural tool for control and data entry is still the keyboard.

Mobile machines offer little surface space, so only a few buttons are available for the entry of text. The most representative form is the use of 10 keys for text entry on mobile phones. However, even smaller machines continue to appear, such as watch-sized computers. It might not be possible to equip such machines with more than four or five buttons. Questions then arise. Is it possible to enter text with a small number of buttons? What about four buttons? How efficient can we make this?

Other potential applications for text entry with four buttons include the control panels of office machines and household machines. Although these machines increasingly contain functions that allow access to the Internet, sufficient surface space for a full keyboard is often not available. Another potential application is in text-entry interfaces for elderly and disabled people. A report (of Advanced Design of Integrated Information Society, 2000) indicates that keyboard operation is the highest hurdle to the use of computers by the aged. The situation is even worse for people with hand-related disabilities. A text-entry device with four large buttons might facilitate human-machine communications by such people.

The idea of decreasing the number of keys on the keyboard in itself is not new. The oldest realization of this idea is the stenotype keyboards. With the recent popularity of mobile machines, researchers have become increasingly interested in one-handed keyboards(Mathias et al., 1996). Most of the work to date in this field has been related to mobile phones. Text entry on current-generation devices remains cumbersome, so innovative companies(Tegic 9, 2000)(ZI-Corp., 2000)(Slangsoft, 2000) have proposed predictive methods for the more efficient entry of text by implementing a method that had first been proposed some years earlier(Rau and Skiena, 1994). The results of several studies have verified its efficiency(James and Reischel, 2001)(Tanaka-Ishii et al., 2000), so the technology looks promising in the context of mobile phones.

Our study goes further in decreasing the number of buttons than the above-cited studies. In our study, we tried various text-entry methods and found the predictive method to be the best. As far as we know, no other study that includes the application of a language model has yet been carried out in this context; neither has the efficiency of this approach been examined. Additionally, the major contribution is our study of the potential power of a language model by examining its actual efficiency on a device with few buttons.

In the next section, we first show how text is entered via our TouchMeKey4 keypad.

2 An Example

Figure 1-1 shows the GUI for the TouchMeKey4 keypad. Nine buttons are visible, with four on either side of the central boxes plus a 'quit' button on the right-hand side. In this paper, we only count those buttons that are only used for the entry of characters that is, the four on the right-hand side. We also impose the constraint that the buttons may only be pressed one at a time, because the inclusion of key-chords increases the actual number of buttons by including the combinations of keys.

Six or seven letters of the alphabet are assigned to each of the buttons. The no. 1 key has 'abcdef', the no. 2 key has 'ghijkl', the no. 3 key has 'mnopqrs', and the no. 4 key has 'tuvwxyz'. The small letters

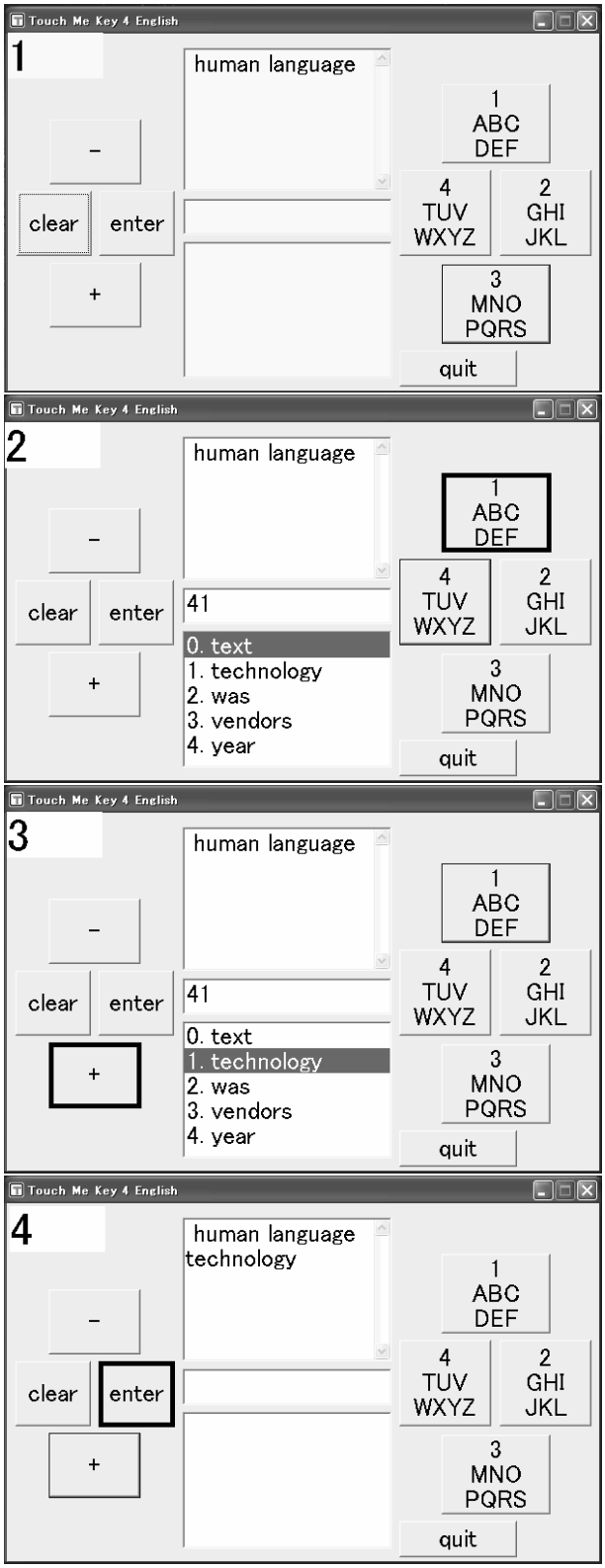

Figure 1: Entering the word "technology" with the TouchMeKey4 keypad

are assigned to the same keys as the corresponding capital letters. All other ASCII characters other than the alphanumeric characters are assigned to the no. 4 key.

Suppose that we have just entered the string 'human language'. The text appears in the upper box in the middle of the window (the upper text-box in Figure 1-1). We now wish to enter the word 'technology'. Words are entered through a *single-tapper-character* form of predictive entry; a key is only *pressed once* to enter a character. For example, the no. 4 button is pressed once to enter the 't' of 'technology'. To enter the subsequent 'e', the no. 1 button is pressed once.

After the no. 1 button has been pressed, the TouchMeKey4 window is as shown in Figure 1-2[1]. Here, we see two differences from Figure 1-1. The first is that '41' appears in the box in the middle of the window. This indicates the string that the user has just entered. The second change is that some words have appeared in the lower box in the middle of the window (a list-box that we call the 'candidate-box'). These words are the candidate words that correspond to the user's input, '41'.

Each press of a button by the user makes the TouchMeKey4 system automatically search the dictionary for candidates. The candidates include longer words as well as, if such words exist, words of the same length as the entered sequence of digits. The candidates are thus all words that begin with one letter from 'tuvwxyz' followed by one letter from 'abcdef'. For example, 'text', 'was', and 'vendors' are candidates, as is the two-character candidate 'we'.

The numerous candidates are sorted into an order before they are placed in the candidate box and shown to the user. The order is according to word probability as determined on the basis of PPM (prediction by partial match), which has been proposed in the information-theory domain. A detailed description is given in §4, but we summarize the method's essence here as part of our explanation of Figure 1. The relevance of each candidate is measured by statistics from two sources.

Base dictionary the unigram statistics collected from a huge corpus of newspaper data, and

User corpus the ngram statistics obtained from a small personal document supplied by the user.

In this example, the Base dictionary is constructed from one year of issues of the Wall Street Journal (WSJ) that contains 93 000 different words and the User Corpus is a computer magazine that contains 10 000 words. The particular User corpus is the reason for the appearance of the relatively uncommon word 'vendors' among the top five candidates (Figure 1-2).

Our target 'technology' appears as the second-ranked candidate. In selecting this word, the user highlights it by using the down button on the left-

[1] Note that the most recently pressed button is framed by a thick line.

hand side of the window(Figure 1-3) and then presses the enter button (Figure 1-4). We see that the selected candidate now appears in the upper text-box[2].

In describing our realization of the TouchMeKey4 system outlined above, the following four questions are discussed in the remainder of this paper:

Interface Is some method other than that described above suitable for text entry with a four-button device?

Candidate Estimation How can the system estimate the relevance of each candidate?

Key Assignment How should characters be assigned to the individual buttons?

Number Of Keys What is the minimum number of keys required? Is the entry of free text with only two buttons reasonably efficient?

3 Interface

Various methods for the entry of text via a four-button device are conceivable. The biggest choice is whether or not to adopt a predictive method.

3.1 Non-Predictive Entry Methods

Let's start by considering the case where we don't adopt prediction. This means that we need to enable the exact entry of the individual characters via the four buttons. One method of this type involves assigning an order to the characters on each key; a key is then pressed i times to obtain the i-th character (we call this the *multi-tap* method). This method is commonly applied on mobile phones.

However, there are two problems with this method. Firstly, the user often needs to press a key numerous times to obtain a single target character. Secondly, there is an ambiguity in the user action when two characters assigned to the same button are to be entered one after another ('aa' requires the entry of '11' that can also be 'b'). This situation requires the use of an escape.

A second possible method is to press a first button to select it, and then enter the number i to select the ith character which is assigned to the first button. For example, on many mobile phones, 'o' is obtained by pressing the no. 6 key and then the no. 3 key, since 'o' is the *third* letter on the no. 6 key. However, if the number of letters on each key is greater than the number of keys, entry of the higher i values is implausibly difficult. With the TouchMeKey4 system, for example, a system for the easy entry of fifth and sixth characters, etc., is not possible.

In short, the free entry of text turns out to be too difficult with a four-button device unless we adopt prediction. This is so even for the case of English, the written form of which has relatively few characters, and is even more so for languages with large numbers of characters such as Chinese, Japanese, or Thai (78 characters). We are thus obliged to use prediction.

[2] As with any system where a predictive method is applied, the weak point of TouchMeKey4 is the processing of unknown words which do not appear in the dictionary. Therefore, it is important that the Base dictionary contains a rich vocabulary. When, however, an unknown word occurs, it may still be entered character by character by using the methods described in§3.1, or the system may be connected with a larger dictionary via a network.

Table 1: Data used in this work

name	WSJ	ZIFF	JA
usage	base dictionary	user corpus	user corpus
domain	newspaper	computer magazine	scientific paper
Total no. wrds (million wrds)	6.6	6.2	9.0
No. diff. wrds (thousand wrds)	93	99	173
Wrds in common with Base Dictionary (%)	100	40	18
Wrds: Avr. len. (L_{avr})	4.45	4.71	4.41
Test document (no. wrds)	-	1900	1826
No. diff. wrds in test doc.	-	868	761

3.2 Predictive Text entry

Generally, there are two ways to predict candidates.

The first is the *single-tap* method. The earliest appearance of this idea was at the beginning of the 80's in Japan, in discussions of processing systems for Japanese text(Co.Ltd., 1982); more recent work has been concerned with mobile phones (James and Reischel, 2001)(Tanaka-Ishii et al., 2000).

The second way is prediction by $prefix$. Given a user input, the system searches for words with the corresponding $prefix$.

This method of collecting candidates to be offered to the user has been particularly successful in the entry of Chinese text. The method has also been applied to certain text-entry systems in the man-machine interface domain, too (Masui, 1999).

As the description of §2 indicates, the combination of the two methods is adopted in our TouchMeKey4 system. It thus needs to process many candidates for a single user entry. The mechanism of estimating levels of relevance for the words is explained in the next section.

4 Applying an Adaptive Language Model in Candidate Estimation

As was summarized in §2, the PPM (prediction by partial match) framework is used by TouchMeKey4 to estimate the relevance of candidates. Its characteristic is that the word distribution is adapted to the style of the user's corpus.

PPM was originally proposed as an adaptive language model for use in improving the compression rates of arithmetic coding. The estimation of probabilities by PPM thus guarantees a lowering of the

entropy of the language model. PPM has successfully been adapted to the user-interface domain in certain previous works(Tanaka-Ishii et al., 2001)(Ward et al., 2000).

Broadly, PPM interpolates the n-gram counts in the user corpus and the statistics in the base dictionary. The following formula is used to estimate a probability for the ith word w_i, $P(w_i)$:

$$P(w_i) = \sum_{k=-1}^{kmax} u_k P_k(w_i) \quad (1)$$

Here, k, the order, indicates the number of words before w_i that are used in the calculation of $P_k(w_i)$. For example, $P_2(w_i)$ is estimated on the basis of the occurrence of w_{i-1} and w_{i-2}. $P_k(w_i)$ is calculated as:

$$P_k(w_i) = \frac{c_k(w_i)}{C_k} \quad (2)$$

where C_k is the frequency of the order k as a context, and $c_k(w_i)$ is the frequency with which w_i occurs in that context. $P_k(w_i)$ when k = -1 describes a base probability that is obtained from the base dictionary. For other k, $P_k(w_i)$ is calculated from statistics obtained from User corpus. Finally, u_k is a weighting probability that is multiplied to $P_k(w_i)$ to obtain the final $P(w_i)$. Of the many studies of u_k(Teahan, 2000), we have chosen PPM-A(Bell et al., 1990), the simplest, because our preliminary experiments showed no significant difference in performance among the methods we tried.

We decided to utilize this PPM framework because the context is the most suitable item of information for the elimination of irrelevant candidates. Small machines are in a personal context, and office and household machines are used in particular contexts. With this method, the language model is adaptable on the fly. This is achieved by simply accumulating the user's newly entered text at the end of the user corpus.

In this paper, the Base dictionary contains the unigram probabilities obtained from Wall Street Journal as was explained in §2. We prepared various User corpora,: three in English, three in Japanese and two in Thai. Of these, the characteristics of two of the English User corpora that are used in §6 are given in Table 1.

5 Key Assignment

The assignment of characters to the respective buttons is one determinant of the efficiency of text entry. For example, if all characters from 'a' to 'w' are assigned to the first key and 'x', 'y', and 'z' are respectively assigned to the second, third and fourth keys, the performance in word prediction will clearly be bad. The problem of key assignment remains even when we have eliminated such extreme possibilities, because there are many plausible assignments. We thus need to be able to measure the performance of a key assignment.

Table 2: Key assignments and entropy

N_K	Lab.	Groups of characters	Entropy
10	-	$(S_0)(S_1)$(abc)(def)(ghi)(jkl) (mno)(pqrs)(tuv)(wxyz)	0.73
5	**A**	(abcdef) (ghijkl) (mnoS_0) (pqrsS_1) (tuvwxyz)	1.09 1.09
5	**B**	(S_0S_1) (abcdef) (ghijkl) (mnopqrs) (tuvwxyz)	1.40
5	**C**	(S_1mno) (abcpqrs) (deftuv) (ghiwxyz) (jklS_0)	1.14
4	**A**	(abcdef) (ghijkl) (mnopqrs) (tuvwxyzS_0S_1)	1.40
4	**B**	(S_1abc) (defghi) (jklmnopqrs) (tuvwxyzS_1)	1.56
4	**C**	(S_0jkl) (abcmno) (defpqrstuv) (ghiqxyzS_0)	1.76
3	**A**	(S_1abcdef) (ghijklmno) (pqrstuvwxyS_0)	1.95
3	**B**	(S_1ghipqrs) (abcjkltuv) (defmonqxyzS_0)	1.91
3	**C**	(S_1jklpqrstuv) (abcdefmno) (ghiwxyzS_0)	2.13
2		(S_0abcdefghijkl) (mnopqrstuvwxyzS_0)	3.28

One way to measure this is to experimentally decide it by automatically entering some documents (as will be described in the §6 later in this paper). However, the result of such a test is dependent on the test document which is used. Lower-level settings, such as key assignments, should, as much as is possible, be for general-purpose use.

Having key sequences as C and the target word as W, the task of the system is to estimate a better W from C. Information theory provides us with a tool for estimating the uncertainty of this task: *the average conditional entropy*. The definition of this quantity, $H(W|C)$, is given by:

$$\begin{aligned} & H(W|C) \quad (3) \\ = \; & \Sigma_{w,c} P(C=c) H(W=w|C=c) \\ = \; & -\Sigma_{w,c} P(C=c, W=w) log P(W=w|C=c) \end{aligned}$$

where $P(C = c)$ is the probability of the input sequence c and $P(W = w|C = c)$ is the conditional probability of words for the given c. When the estimation of W is less certain, $H(W|C)$ has a larger value. The lower the entropy, the less uncertain the estimation of the word. Therefore, the conditional entropy is suitable as a method for the evaluation of key assignments.

One other factor that we need to consider at this point is the order of the alphabet. English has an alphabet order that even children know. If this order is neglected and the letters 'ajxgukh' are assigned to a given key, the interface will become difficult for the beginners, although it might be the most efficient for

a well-trained user. Therefore, the key assignments had better reflect such linguistic tradition.

We took this into consideration in generating some possible key assignments. Table 2 is a list of the assignments and their values of conditional entropy as calculated on the basis of one year of issues of WSJ. The first column shows the total number of keys (below denoted by N_K). We here consider the situations where there are five, three, and two, as well as four, buttons. The second column gives a label for each of the key assignments. In the third column, the characters to be assigned to the respective buttons are grouped in parentheses. For example, 4-**A** indicates an assignment to four keys with 'abcdef' assigned to the first button, 'ghijkl' to the second button, 'mnopqrs' to the third, and 'tuvwxyz' and other ASCII symbols to the fourth. The capital letters are assigned to the same keys as the corresponding small letters. S_0 and S_1 indicates the non-alphabetic ASCII symbols [3]. Note that 4-**A** corresponds to the TouchMeKey4 assignment which we saw in Figure 1-1. The groupings with the label **C** are more random than those with other two.

In general, entropy values fall as the number of keys increases. This is a readily comprehensible result; a larger number of keys eases the task of estimation, thus making it less uncertain. When we compare the values for assignments to the same numbers of keys, we see that the entropy values differ considerably. For example, the entropy of 5-**B** indicates more uncertainty than the other 5-x assignments. The entropy value is the same as for 4-**A**, although the number of keys in use is different (this is comprehensible when we look at the similar character groupings of 4-**A** and 5-**B**).

In this paper, we evaluate the use of key assignments with the label **A** on TouchMeKey4, since they have lower entropy values than the other settings.

6 Evaluation

6.1 Number of Keystrokes

We attached an automatic text-entry routine to TouchMeKey4 and measured the numbers of keystrokes that are needed per word. The number of keystrokes is the sum of the keystrokes required for the input and selection operations. Keystrokes for selection are counted to be n when choosing the nth-candidate.

In the prediction of words, there are multiple points where the target word may be chosen. For example, in Figure 1, the word 'technology' appears as the second-best choice after the user has typed in '41'. The user may select the target at this point or type in another '1' to indicate the 'c' and increase the target's rank. The automatic routine only chooses the target after it has appeared as the best candidate; otherwise, the routine continues to enter the word. When the full length of the word has been entered, the target word of the current ranking is chosen.

[3] These symbols are categorized into two groups according to the categorization used on mobile phones.

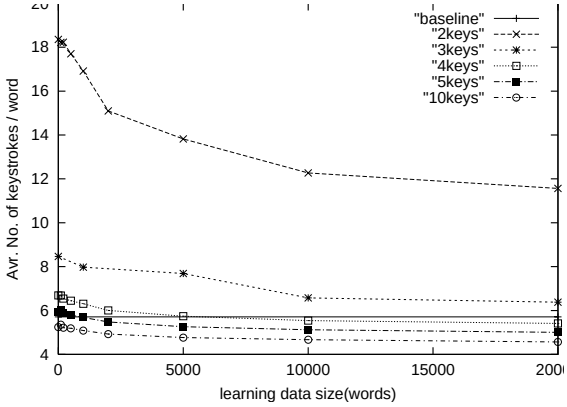

Figure 2: No. keystrokes with learning of ZIFF

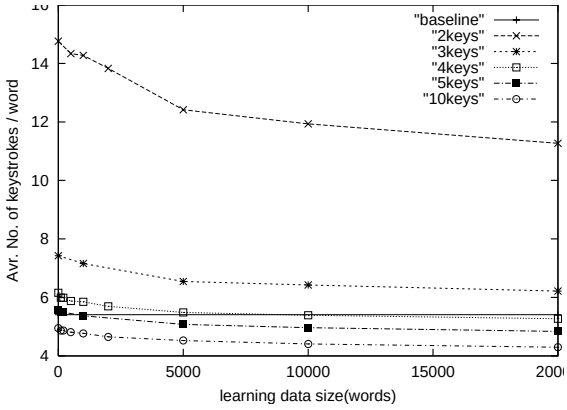

Figure 3: No. keystrokes with learning of JA

Figure 2 and 3 show the relation between the amount of learning data (horizontal axis) and the number of keystrokes per word (vertical). The test document is indicated in Table 1 (7th row). Data of the same kind but from different positions in the test data are used for the learning data and the test data.

The respective lines indicate learning when N_K is 10, 5, 4, 3, and 2. The larger the N_K, the lower the line. The horizontal solid line (around 5.5 taps per word) indicates the baseline, the average number of keystrokes needed to process a single word on a full keyboard. This is calculated as L_{avr} (in Table 1)+1(for space). Note that TouchMeKey4 automatically enters the space.

When there is no learning data, TouchMeKey4 needs far more keystrokes than the baseline. However, after the learning of ten thousand words, the number of keystrokes goes below the baseline when $N_K > 4$.

In order to see the results at macroscopic scale, Table 3 shows the results after the learning of 50 thousand words. The values indicate per-word keystrokes, and the percentages in the parentheses show the ra-

Table 3: No. keystrokes per word with learning of 50 000 words of the user corpus

total number of keys (N_K)	ZIFF	JA
10	4.36(16.7%)	4.15(16.1%)
5	4.83(18.9%)	4.84(13.2%)
4	5.19(22.4%)	5.06(17.8%)
3	5.97(29.5%)	6.08(19.1%)
2	11.87(35.4%)	10.63(28.0%)

tios by which the numbers of keystrokes decrease as compared with the case of no learning of a user document. We see that the numbers of keystrokes are reduced by about 30% for both ZIFF and JA. When $N_K = 4$, the value falls to around 5.1. Since L_{avr}+1 is around 5.5, TouchMeKey4 provides a reduction in numbers of keystrokes of almost 9 % as compared with a full keyboard. Superficially, this looks like a small gain. However, it is surprising that, even with 4 buttons, text may be entered with *fewer keystrokes* than with a full keyboard.

When $N_K = 3$, on the other hand, the number of keystrokes remains at around 6.0 per word. Therefore, when N_K=3, the system requires a larger number of keystrokes than the baseline.

TouchMeKey4 also runs in Japanese and Thai, so we executed analogous experiments with those languages. We obtained very similar graphs in these cases. To resume, here are our observations across three languages:

- Learning is indispensable for systems with small N_K values to perform better than the baseline.
- However, a large amount of learning data is not necessary (text with ten thousand words is enough).
- When $N_K = 3$, the number of keystrokes does not fall below the baseline.

6.2 Speed

Eight subjects were hired to test TouchMeKey4: three in English, three in Japanese, and two in Thai. Two of the subjects for the English are native speakers of Japanese. The other subjects were the native speakers of the languages in the respective tests.

The subjects were told to do 10 sessions of testing. Each session is 30 minutes long; the subject was told to continue to enter the given text as quickly as was possible and without pausing during each of the sessions. The vocabulary of the given text is solely from the learned user corpus. TouchMeKey4 learned a 10-thousand-word user corpus before it was handed to the subjects. For the text entry, they were given hardware controllers that work with TouchMeKey4.

Figure 4 gives the results on speed. The horizontal axis describes the sessions and the vertical axis shows the average numbers of words per minute (wpm) in each session. The respective lines indicate the speed

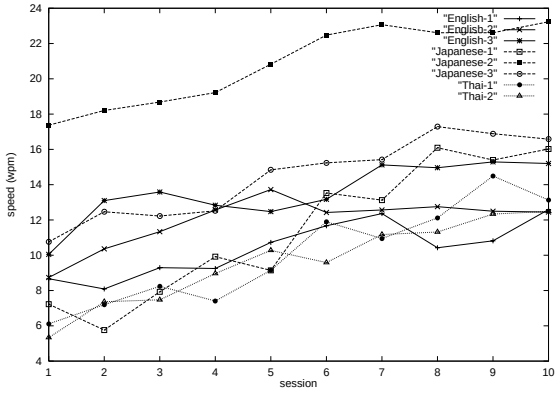

Figure 4: Speed

of the subjects over time. After 5 hours training, entry by each of the subjects was at some rate above 12 wpm. The speed of entry by the multi-tapping method on a mobile phone is in the range from 5 to 10 wpm(James and Reischel, 2001), so TouchMeKey4 obviously allows higher rates of text entry. Furthermore, the speeds are comparable to those obtained with the single-tapping method on mobile phones (7 to 25 wpm(James and Reischel, 2001)). One subject set the record, reaching 23 wpm.

This speed is comparable to that of an expert with the single-tapping method. Predictive text entry thus prevented deterioration of performance, despite the number of buttons being decreased from 10 to 4.

With regard to human learning, the more highly the subject was trained, the faster he or she became. The speeds of some subjects who had had difficulties at the beginning of the tests had doubled by the end.

Language-by-language comparison reveals that Japanese text entry was fastest. Although the entropy value for Japanese is by far greater than the values for Thai and English (4.05 for Japanese and 1.13 for Thai), the Japanese subjects managed well with TouchMeKey4 because they are accustomed to the use of predictive text entry in kana-kanji conversion.

We must admit that TouchMeKey4 places a heavier cognitive load on users than does text entry via a full keyboard (40 to 60 wpm) or a stylus and virtual keyboard (32.5 wpm(Zhai et al., 2000)). However, we regard the speed as satisfactory in comparison with those achieved by using single-tap-by-character entry systems on mobile phones.

7 Conclusion

We have presented TouchMeKey4, a text entry device that requires only four buttons, and aspects of its design and testing. Several characters of the alphabet are assigned to each of four buttons and the user enters a document with the aid of a predictive text-entry system. The device is realized by software that estimates the word that most probably corresponds with

the user input. The estimate is based on an adaptive-statistical language model. Firstly, a base model is constructed from a large body of text from newspapers; the model is then adapted to the local context by using a smaller user corpus of 10 thousand words.

Our evaluation shows that text entry with this system is as efficient as text entry on a mobile phone. The number of keystrokes is reducible to a number below that required on a full keyboard. The average speed of our test subjects was 14 words per minute, and the fastest subject recorded 23 wpm. We also discussed the possibility of entering text via even fewer keys, e.g., three keys. However, the usability of a three-key device turned out to be questionable, because the number of keys needed to enter text becomes large.

Our future direction will be to investigate the actual application of the system on smaller machines and to develop a text-entry system for use by elderly and disabled people.

References

T.C. Bell, T.G. Cleary, and T.H. Witten. 1990. *Text Compression*. Prentice Hall.

Toshiba Co.Ltd. 1982. Japanese entry system. In *Japanese Patent 57-185528 (in Japanese)*.

C. James and K. Reischel. 2001. Text input for mobile devices: Comparing model prediction to actual performance. In *Proceedings of the Annual SIGCHI Conference*.

T. Masui. 1999. Po_box an efficient text input method for handheld and ubiquitous computers. In *the ACM Symposium on User Interfface Software and Technology*, pages 113–119.

E. Mathias, I.S. MacKenzie, and W. Buxton. 1996. One-handed touch typing on a qwerty keyboard. In *Human Computer Interaction*, volume 11, pages 113–155.

Research Association of Advanced Design of Integrated Information Society. 2000. `http://www.pref.toyama.jp/sections/1012/matsplan/kadai/k10in`.

H. Rau and S. Skiena. 1994. Dialing for documents:an experiment in information theory. In *UIST*.

Slangsoft. 2000. Slangsoft home page. `http://www.slangsoft.com`.

K. Tanaka-Ishii, Y. Inutsuka, and M. Takeichi. 2000. Japanese input system with digits–Can Japanese be input only with consonants?—. In *Human Language Technology Conference 2001*.

K. Tanaka-Ishii, Y. Inutsuka, and M. Takeichi. 2001. Personalization of text input systems for mobile phones. In *NLPRS*.

W.J. Teahan. 2000. Probability estimation for ppm. In *NZCSRSC'95*. `http://www.cs.waikato.ac.nz/ wjt/papers /NZCSRSC.ps.gz`.

Tegic 9. 2000. Tegic 9 home page. `http://www.t9.com`.

D. Ward, A.F. Blackwell, and D.J.C. MacKay. 2000. Dasher:a data entry interface using continuous gestures and language models. In *the ACM Symposium on User Interfface Software and Technology*, pages 129–137.

S. Zhai, M. Hunter, and B.A. Smith. 2000. The metropolis keyboard- an exploration of quantitative techniques for virtual keyboard design. In *UIST*.

ZI-Corp. 2000. ZI home page. Available from `http://207.229.18.241/`.