Japanese Text Input System With Digits

—Can Japanese text be estimated only from consonants ?—

Kumiko TANAKA-Ishii Yusuke INUTSUKA Masato TAKEICHI
University of Tokyo
7-3-1 Hongo Bunkyoku Tokyo
+81-5841-7412, Japan
{kumiko, inu, takeichi}@ipl.t.u-tokyo.ac.jp

ABSTRACT

We discuss a Japanese text input method for mobile phones. Different from the current methods that are based on kana kanji conversion system, our system asks user only to input the sequence of digits that in fact corresponds to the sequence of consonants, and then convert it directly into the final kanji form.

After the examination on the number of candidates for a sequence of digits, we explain our method of word estimation based on Hidden Markov Model and describe our implementation. Then we report the results of evaluation by comparing the number of needed keystrokes to input various kinds of actual texts.

1 INTRODUCTION

As the mobile phones have come into wide use, various internet services are now becoming available on these phones. For example, we may look up train timetables, or we may ask for the best Italian restaurant around the place. This is usually done not by calling up somebody, but by accessing a given home page on the phone, and by typing in the needed text.

The problem here is that mobile phones are only equipped with a small keypad of about 12 small buttons. Therefore, people are now making great efforts to minimize the number of keystrokes to input text. This is done not only from the services side (such as to use pop up menu for selection), but also from the users' side (such as to register the frequently used keywords as shortcuts). Additionally, there is another trend that appeared recently, that is, to adopt the new input method, *single-stroke-per-character* method [9][10].

With the past text input method, users have to press the keys multiple times to obtain a suitable character (*multiple-strokes-per-character* method). For example, to input "box", user taps the sequence of "2266699". Here, 'b' corresponds to the No.2 key assigned three characters of "ABC", so, "22" means the *second* character "B". Similarly, tapping No.6 three times indicates the third character of "MNO" and twice No.9 is the second character of "WXYZ". Rather, single-stroke-per-character method proposes only to press "269" for "box", the first digits of the same successive numbers. Off course this sequence does not only correspond to the word "box" but also to, "any", "cow", "boy" etc. So, the system picks all these possibilities, show them to the user and the user chooses his target word. In essence, single-stroke-per-character method tries to decrease keystrokes by increasing the ambiguity of input and resolving it interactively.

In Japanese, single-stroke-per-character method corresponds to input digits that actually are consonants. In the early 80's, when the current kana kanji conversion system was still not that established, the discussion of Japanese input system only by consonants existed in the context of desk-top computer applications [3][4]. Shortly after, the research extinguished without being able to find any application. However, the recent spread of mobile phones has once again brought this topic to the fore.

In general, single-stroke-per-character method needs less number of key strokes for input, but it should handle far larger number of candidates compared with multiple-strokes-per-character. Additionally, this problem of explosion of candidates is greater in Japanese, first because the Japanese language contains more homonyms and second because the kana alphabet has twice as many characters as the European alphabet.

Therefore, we first need to verify whether such system could be put to use. After over-viewing the current Japanese input method with digits, we will explore the possibilities for the containment of this explosion of can-

didates.

2 INPUT WITH DIGITS

Before we go into our topic, let us briefly explain the kana-kanji conversion system, that is used nowadays to input Japanese text on desktop computers.

The final form of Japanese could be the mixture of two sort of characters; *kanji*, the Chinese character, *kana*, the phonetic alphabet character. We input such a language as follows. First, the user inputs his target word by the sequence of phonemes, using *kana*. Then, the input system looks for the possible final forms that correspond to the user input and show them to the user. The input finishes when user selects his choice.

Currently. there are two major methods to do the same with digits. In both methods, users input the sequence of kana by digits. It is the user's responsibility to convert kana into digits. Because a kana is composed of a consonant and a vowel, each of them is attached a digit as is shown in Table 1. The user may convert any kana sequence into a unique sequence of digits with this table. For example, "は" (ha) corresponds to "61" and "こ"(ko) to "25" .

The two input methods differ in how users input the vowel (see Table 2[1]). The first method is called *Pocket-Bell method*[2] (written as PB method in the followings). The user inputs the digit sequence exactly as was obtained from the Table 1. For example, to input "はこ" (reads *hako*, means *box*), user presses "6125" successively. This method has the priority that the user needs only to input two digits per kana just as with the ordinary keyboard on desktop computers. However, the user needs to move his fingers more than the following second method.

The second method is called *Kana method*(with capital 'K'), the most popular input method among the mobile phone users at the moment. Users tap consonant digit *the vowel digit times*, just as in the multiple-strokes-per-character method for European language. In the case of "はこ", users tap once No.6 key, then five times of No.2 key. This method has a priority that the user could keep his finger at one place to input one kana. However, he needs to make many key strokes, especially when the word contains kana with the vowel of "e"(four times) or "o" (five times). Because of this, the biggest problem of users in this method is that they tend to mistype by pressing the same button too many times than expected. Additional problem is that there is an ambiguity of man-machine interaction when users want to input the kana of the same consonant successively. For example, "hihi" (6262), 6 twice and then 6 twice again is exactly the same as four times of 6 (64) that corresponds to "he".

[1]Here "*" is used to transfer "k" to "g", that is similarly written in kana alphabet. For example, with the vowel "o", "こ"(ko) is transformed into "ご" (go). For PB method, 1 is needed after "*" because there are several transformation of kana of this kind. Also, "ん"(n), the unique character in Japanese without vowel is input by "03".

[2]Pocket Bell is NTT's trademark for pager.

Table 1: Digits corresponding to consonants and vowels

Consonants

0	1	2	3	4
わ (w)	あ (ϕ)	か (k)	さ (s)	た (t)
5	6	7	8	9
な (n)	は (h)	ま (m)	や (y)	ら (r)

Vowels

1	2	3	4	5
a	i	u	e	o

Table 2: How "にほんご"(reads *nihongo*, means *Japanese*) is input with digits

	に	ほ	ん	ご
PB method	52	65	03	25*1
Kana method	55	66666	000	22222*
Our method	5	6	0	2*

We could observe that what users want is a method that allows text input with:

- less number of replacement of fingers, and
- less total number of keystrokes.

Our method asks user to input one stroke of the digit that corresponds to the consonant of kana. In the case of "はこ", the user only needs to input the sequence of "62" (corresponding to "hk"). This method has a priority that user needs only one stroke per kana, and he neither needs to move fingers for a kana. The number of strokes by the user decreases to the half that of PB method.

Instead, because such input sequence is more ambiguous compared with the kana-based method, the system needs to guess the actual user input among candidates that have the same sequence of consonants. For example, "62" not only corresponds to "はこ", but also to other completely different kana sequences such as "ふく"(reads *huku*), "はか"(reads *haka*). Further, for each of these kana sequences, there are several final forms. For example, "ふく" corresponds to, "服" (means *dress*), "吹く"(means *to blow*), "拭く"(means *to wipe*), "福" (means *blessing*), "副"(means *vice*) etc. Therefore, the number of candidates is expected to be very numerous compared with kana-based method. In the next section, we examine some statistics and discuss whether our input method could be put to use.

Table 3: Number of candidates for a digit sequence

	statistics	Our method	Kana or PB method
Base line	Average	2.41	1.39
	Max	167	43
	Average by frequency	31.61	4.76
Words with POS tag	Average	1.69	1.19
	Max	74	37
	Average frequency	5.47	1.65
Words of frequency more than 100	Average	1.82	1.15
	Max	32	10
	Average by frequency	6.29	1.75

3 PROBLEM

Japanese corpus of Mainichi newspaper articles of general news page ('94, 1.3 million word occurrences, 28Mbytes) contains 90 thousand different words. The average length of words when transliterated in kana alphabet is 4.81. Because kana has 50 different characters, $50^{4.81}$ = about 143 million different kana sequences can be represented with the length of 4.81. Among 143 million, only 90 thousand are used for the vocabulary of Mainichi newspaper. Therefore, one kana sequence per one word can be easily realized if we do not think about homonyms.

On the other hand, our method only has 10 digits (consonants). This can represent only $10^{4.81}$= about 60 thousand different digit sequences. In this space, we should assign 90 thousand words, that is larger than the space size. Therefore, large number of words need to share a digit sequence even if all different digit sequences are used. Here already we have a hunch that the number of word candidates for a sequence becomes very large.

To measure the number of candidates more precisely, we took the statistics of the number of candidates given a sequence of kana or digits that corresponds to a word. The first line of Table 3 shows the average number of candidates for an input in the case of our method and by PB or Kana. We could already see that in our case, the number of candidates amounts to more than 2 taking any word at random. The second line shows the maximum number of candidates; our case is four times as much as kana's case. Additionally, we also calculated the average number of candidates taking a word according to the word frequency distribution[3]. Looking at this line, we nearly feel that we should give up this problem.

One solution to handle this explosion of candidates is to make the selection process into two stages rather than one. After input, the user is first shown to choose among possible kana sequences. For example, when he inputs "62", then the system first shows "はか", "はき" (reads *haki*), "はく",..., " ほこ"(reads *hoko*) and user chooses his target as kana alphabet sequence. Then, the user's choice is passed to the second selection process, the kana-kanji conversion system. Such solution is taken by T9[8], or ZI[10].

However, we have a strong impression that such selection process had better be unified if possible. The largest reason for this is that twice of selection process makes the man-machine interaction rather complex. Additionally, the action of interactive candidate selection is slow because the user should look for his target by scrolling the candidates back and forth[4]. Therefore, the number of selection process had better be eliminated as much as possible.

Consequently, we seek to implement our input method within single stage of selection as is in the kana-kanji conversion system. User inputs digit and our system converts it into the final form directly. In this case, we should do something with the number of candidates. There are only two solutions: 1.decrease the whole number of candidates and 2. sort them in a preferred order.

In order to decrease the number of candidates, we may use the part of speech tag in order to discriminate words better. The 4th to 6th lines in Table 3 show the same statistics for the whole words but discriminated with part of speech tag. We could see that the values decrease.

The naive criteria to sort candidates in a better order is the frequency. Suppose that there are 32 candidates, 2 of them are frequently used but the rest 30 are hardly used. Then if the system sorts the candidates in the frequency order and shows that two as the best, then the user need not handle the useless 30 candidates.

In order to see whether frequency could be used or not, 7th to 9th rows in the Table 3 shows the same statistics calculated only for the words that occurred more frequently than 100. We could see that the number of candidates decreases to better values. Therefore, we could say that frequency information helps to show the candidates in right order.

To conclude, we try to implement our Japanese input

[3]This is done by the following formula.
$$\frac{\sum(number\ of\ candidates)\times(freqency)}{\sum frequency}$$
Note that sum is calculated for all possible kana/digit sequences.

[4]In Japanese, we also had character based input method named *T-code* system, that is proved to be the far faster method to input Japanese, compared with any kana-kanji conversion systems. A T-code is two successive keys that corresponds to one character. T-code users first memorize codes for all Japanese characters that amounts to more than 5000. Although proved to be the fastest input method, the load of memorizing T-code was too tough to be accepted by most of the end-users.

Table 4: Various text input options

Name	Input	Unit	Completion	Selection Stage	Current System
DW1	digit	word	no	one	-
DW2	digit	word	no	two	T9[9], ZI[10]
DWC1	digit	word	yes	one	ours
DP1	digit	phrase	no	one	ours
KWC1-PB	kana(PB)	word	yes	one	-
KP1-PB	kana(PB)	phrase	no	one	any mobile phone input system
KWC1-Kana	kana(Kana)	word	yes	one	PO-Box[5] (for Palms)
KP1-Kana	kana(Kana)	phrase	no	one	any mobile phone input system

system that allows user to select the final target within the **single selection stage**. In order to do this, we make much use of the **part of speech tag** and **word frequency**.

4 LANGUAGE MODEL

The input system needs to estimate the corresponding word sequence from the input sequence of digits. We adopt language model based on Hidden Markov Model for this task.

Suppose that C denotes the user input sequence of digits. Then the best sequence of words is defined as:

$$\hat{W} = \arg\max_{W} P(W|C) \tag{1}$$

Because C is the same to all candidates, right hand side of equation is:

$$\hat{W} = \arg\max_{W} P(W). \tag{2}$$

If we denote T as the sequence of part of speech tags, then $P(W)$ can be rewritten as:

$$P(W) = \sum_{T} P(W,T) \tag{3}$$

without loosing any generality. We introduce two approximations:

$$P(w_n|w_{1,n-1}, t_{1,n}) = P(w_n|t_n) \tag{4}$$

$$P(t_n|w_{1,n-1}, t_{1,n-1}) = P(t_n|t_{n-1}). \tag{5}$$

Here w_i means the ith word of W, $w_{i,j}$ denotes word sequence from ith to jth of W. Then the right hand side of the equation (3) is transformed into

$$P(W) = \sum_{T} \prod_{i=1}^{n} p(w_i|t_i)P(t_{i+1}|t_i). \tag{6}$$

Such a word model for English is resumed by Charniak[2] , and also in Japanese by Nagata [7] especially for morphological analyzer.

Overall, given a digit sequence, the system calculates the above probability for all the possible candidates, and then sorted shown to the user in that order.

5 IMPLEMENTATION

5.1 Input Options

There are other various options to input text other than input by digit vs. kana (second column of Table 4). One is the language unit. The choice are among a word, a phrase or phrases (third column)[5]. Here, the longer the unit is, the larger the system load because it should look for the best target among combinatorially many candidates.

We could also think of whether to adopt the completion (fourth column). When using the completion, the system estimates and shows the best candidate not waiting until the input to be as long as the unit. For example, with DWC1, "箱"(reads *hako*, means *box* corresponds to "65") is estimated and shown to the user even when the user only taps "6". This method is very ambiguous at input. Such a input is proposed for Palm text input systems and it is distributed as free-ware [5] for kana-based input(7th row of Table 4).

Completion by phrase means to estimate the best next word as well as completing the current word. Therefore, the system might estimate " 箱は" (*a box is*) or "箱入り娘は" (*a girl from a good familiy is*, Japanese idiomatic expression using the word *box*) after a single input of "6". In this case, the load on the system will become very high because the search area for candidates is very vast.

The fourth column of the table shows how many selection stages are used until user to end the input. Our choice is one, as is explained §3.

Here, our input system is mainly to input text with the methods of DWC1 and DP1. However, we also implemented other variations as in the list, so that all could be compared against each other.

5.2 System

In general, given a digit sequence, the system looks up all possible candidates using the dictionary (see §5.3)

[5]Character based input means no ambiguity at input.

that is initialized at boot time. Then it calculates the probability (explained in §4) for candidates, sorts them and shows them to the user.

If the user's target word is contained in the system results, then the user may choose the word using one of the following commands:

- **+n** Select the first word of n-th candidate.
- **m** Show the next candidates.

When the input method is phrase-based, then the system cut the input sequence and convert each piece into words. For example, " 421430316 " could be words such as " 武市 (4214, takeiti, *name*) 先生 (3031, sensei, *professor*) は "(6, ha), or "竹内 (4214, takeuti, *another name*) 新政府 "(30316, sinseihu, *new government*). In this case, the system should estimate two parameters: word border and the word itself.

The number of candidates in phrase based method is large because of combinations. So, in order to decrease the calculation complexity, we adopted Nagata's Viterbi-like algorithm[7] to approximately obtain the best candidate. From this best candidate, its first word is replaced with the other possible words of the same digit length. The resulting set is shown in the order of the probability.

We could have chosen to form candidates out of the second best, or the third best according to the formula (6) and Nagata's method. However, the second and the third best could contain words of different border. This forces users to make selection among words of different length, that is rather confusing from user interface point of view. Therefore, we took the above method to form candidates, so that users may decide one parameter at a time, first by adjusting word border and then choosing the target word.

In order to allow users to adjust the word borders, some more commands are prepared:

- **s** Shorten the word border of the first word
- **l** Extend the word border of the first word

These commands are to be implemented using the direction key and special keys that is also equipped on mobile phones.

Another problem that might occur is when the user's target was not found in the dictionary that is *unknown words*. In this case, the user might need to input character by character (see §5.3). If the user's target is written by kana or kata-kana, then user may do this by:

- **h** Transliterate first word with kana
- **k** Transliterate first word with kata-kana

Unknown words will be registered automatically into the user dictionary. (However, for the evaluation section, registration is not performed to measure the system performance in the identical environment.)

For all input methods, human user can be replaced by a routine that automatically inputs any given Japanese text and counts how many keystrokes are needed for the task. Candidates are formed and shown to the routine using exactly the same statistical method of frequency and part of speech tag based on HMM. All methods use the same dictionary (described in the next section). Therefore, the number of key strokes can be compared fairly.

As for methods of completion, the timing to select the candidates are not unique. For example, user might find his target as the third best after the input of 2 digits, or find his target as the best after the input of 3 digits. For the automatic input, we decided that the target word is chosen when it appears as the best candidate. Otherwise, the next digit of the current word is tapped in to filter out the irrelevant candidates. When input for the word ends and the target does not appear as the best, then the target is chosen from the non best.

5.3 Dictionary

The dictionary is constructed from Mainichi newspaper corpus (described in the §3). First, we analyzed corpus morphologically[6], then all words that occurred in the corpus were shaped into a dictionary. One entry of the dictionary contains the following elements:

- word transliterated in kana
- corresponding sequence in digits (only of consonants)
- word
- part of speech tag
- frequency

The dictionary is added all characters in Japanese to cope with the unknown words. In order to minimize the size of the whole dictionary, we implemented the dictionary using trie-based method [1].

6 EVALUATION

6.1 Output Example

We first show two small input examples each by DP1 and DWC1 in Figure 5. The example phrase is described at the top four rows.

We see that a user inputs a digit by digit when using DWC1, and a phrase by digit when using DP1. Then system estimates candidates (five candidates are shown at a time in the example case. Some part of DWC1 is omitted for the sake of space). Then, the user interactively chooses his target. The results could be seen accumulated in front of the prompt ' >' as the input proceeds. For DWC1, the effect of completion could be seen for the words of "busy", that the target appears even when the input is not completed.

Table 5: Output example

Japanese target text	武市先生はいつも忙しい。
Translation	Prof. Takeichi is always busy.
Transliteration in roma-ji	takeichisensei ha itsumo isogashii.
Digit sequence (only consonants)	42143031 6 147 132*31

DWC1	DP1
> 4	> 421430316
1. とう (トウ)	1. 竹内 (タケウチ) 先生 (センセイ) は (ハ)
2. っぽく (ッポク)	2. 武内 (タケウチ) 先生 (センセイ) は (ハ)
...	3. 徳一 (トクイチ) 先生 (センセイ) は (ハ)
> 2	4. 高市 (タカイチ) 先生 (センセイ) は (ハ)
1. 的 (テキ)	5. 武市 (タケイチ) 先生 (センセイ) は (ハ)
2. 突き (ツキ)	> +5
...	1. 先生 (センセイ) は (ハ)
> 1	2. 真相 (シンソウ) は (ハ)
1. 高い (タカイ)	3. 申請 (シンセイ) は (ハ)
2. 近い (チカイ)	4. 酸性 (サンセイ) は (ハ)
...	5. 戦争 (センソウ) は (ハ)
> 4	武市> +1
1. 竹内 (タケウチ)	1. は (ハ)
2. 付き合っ (ツキアッ)	2. へ (ヘ)
3. 逐一 (チクイチ)	3. 府 (フ)
4. 武内 (タケウチ)	4. 費 (ヒ)
5. 突き落とさ (ツキオトサ)	5. 派 (ハ)
> m	武市先生> +1
(*more* command filters out words longer than length 4.)	武市先生は> 147
1. 竹内 (タケウチ)	1. いつも (イツモ)
2. 付き合っ (ツキアッ)	2. 集め (アツメ)
3. 逐一 (チクイチ)	3. 伊丹 (イタミ)
4. 武内 (タケウチ)	4. 頭 (アタマ)
5. 武市 (タケイチ)	5. 逸見 (イツミ)
> +5	武市先生は> +1
武市> 3	武市先生はいつも> 132*31
1. さん (サン)	1. 忙しい (イソガシイ)
2. 者 (シャ)	武市先生はいつも> +1
...	武市先生はいつも忙しい> *
武市> 0	1. 、 (、)
1. さん (サン)	2. 。 (。)
2. 信 (シン)	3. 」 (」)
...	4. 「 (「)
武市> 3	武市先生はいつも忙しい> +2
1. 信二 (シンジ)	武市先生はいつも忙しい。> quit
2. 選手 (センシュ)	
...	
武市> 1	
1. 先生 (センセイ)	
2. 申請 (シンセイ)	
...	
武市> +1	
武市先生> 6	
1. は (ハ)	
2. へ (ヘ)	
...	
武市先生> +1	
武市先生は> 1	
1. 一 (イチ)	
2. 大阪 (オオサカ)	
...	
武市先生は> 4	
1. 一 (イチ)	
2. あっ (アッ)	
...	
武市先生は> 7	
1. いつも (イツモ)	
2. 一万 (イチマン)	
...	
武市先生は> +1	
武市先生はいつも> 1	
1. いう (イウ)	
2. ある (アル)	
...	
武市先生はいつも> 3	
1. 異常 (イジョウ)	
2. いずれ (イズレ)	
...	
武市先生はいつも> 2	
1. 意識 (イシキ)	
2. 遅く (オソク)	
...	
武市先生はいつも> *	
(system estimated the target as No.2 candidate when the input of the word is not completed yet.)	
1. 急ぐ (イソグ)	
2. 忙しい (イソガシイ)	
3. 忙しく (イソガシク)	
4. 忙しかっ (イソガシカッ)	
5. 薄暗い (ウスグライ)	
武市先生はいつも> +2	
武市先生はいつも忙しい> *	
1. 。 (。)	
2. 、 (、)	
...	
武市先生はいつも忙しい> +1	
武市先生はいつも忙しい。> quit	

6.2 Test Data for Automated Input

We prepared two kinds of texts for evaluation (Table 6). The first kind is the Mainichi newspaper articles. For this first kind, two corpus is prepared, the one used for building the dictionary, and the other articles of economics domain not used to build the dictionary.

Table 6: Test data

	Newspaper		Personal text	
	general	economics	e-mail	book
No. of words	1752	1675	1744	1577
No.of diff. words	771	652	528	330
No. sentences	61	53	72	59
No.unknown words	1	24	22	67
No.unknown diff. words	1 1	24 23	22 21	67 28
Avr.len. of words(digits)	2.431	2.866	2.326	2.246

The second kind is from the completely different domain, that is user's personal texts. Here also, we prepared two texts, the third author's e-mail corpus (of 1 year, about 150 thousand words) and his article of text book of functional programming (about 90 thousand words).

In order to eliminate the local bias of context, we took a certain number of sentences randomly from all over the place of each corpus, until the total number of words amounts to more than 1500 words. Here, stops (periods, commas) are also regarded as a kind of word.

The number of unknown words that occurred at input are also shown in the table. This is not included in the number of words(first line), nor to compute average number of keystrokes in the following. Note that unknown words occur not only because it is not registered in the dictionary, but also because of the approximation of search for candidates (see §5.2).

6.3 Kana vs. Digit

Table 7 shows the results for 6 methods for the test data of economics articles. The table contains average number of keystrokes needed for each action of input, adjust word border (only for phrase based method) and selection. For the selection, we assume that it needs n strokes if the correct answer is shown as nth candidate[6].

[6]Readers might indicate that a stroke is not needed to choose the best candidate, because the user could go on to tap the next word directly without any explicit selection action.

As for the methods with completion, this is not true. Target is estimated at every user action, therefore the user should explicitly choose the target even when choosing the best candidate.

As for the methods without completion, the assumption is true. However, without completion, user should indicate the timing to

Table 7: Average number of keystrokes per word needed for each action to input newspaper articles of economics

	input	adjust wrd border	select
DWC1	2.476	-	1.802
KWC1-Kana	6.204	-	1.106
KWC1-PB	4.952	-	1.106
DP1	2.866	0.116	1.903
KP1-Kana	8.088	0.013	1.088
KP1-PB	5.731	0.013	1.088

Table 8: Average number of keystrokes per word for various test data

	Newspaper		Personal text	
	general	economics	e-mail	book
DWC1	4.446	4.278	4.060	4.731
KWC1-Kana	6.683	7.310	6.442	6.802
KWC1-PB	5.414	6.058	5.260	5.355
DP1	4.864	4.885	4.521	5.268
KP1-Kana	7.993	9.189	7.618	7.735
KP1-PB	5.991	6.832	5.748	5.765

The average total keystrokes needed per word is listed in the second column (of economics) in Table 8.

First of all, we see that PB and Kana methods have the same number of strokes for adjusting word border and selection. This is always true because user ultimately input kana both with PB and Kana methods Also, the number of keystrokes needed for PB method is double that of our method. We could also see that DWC1 needs less number of keystrokes for a word than DP1, that is the effect of the completion.

With DWC1 and DP1 less keystrokes are needed for input, but more for selection compared with KWC1 and KP1 methods. However, as a whole, less keystrokes are needed(Table 8, second column). We see here that our approach is successful for this test document. We also see that DWC1 is the most efficient method.

6.4 Difference among Text Kinds

Next, we compare the efficiency to input text of different kinds. This time, we only show the total average keystrokes per word in Table 8, because the breakups of keystrokes have the same trend as were discussed in the previous section.

invoke the digit/kana kanji conversion, therefore one extra stroke is always needed per language unit which is not counted in our evaluation. Therefore, balancing these two, for DW1, the count is equal to the minimal number of keystrokes. For DP1 and KP1, the counts are slightly larger than the minimal.

For all four texts, we see the same trends as we have seen with economics article, that is:

- Input by digits is more efficient than that by kana.
- DWC1 is the most efficient.

Therefore, we could probably say that, in general, the input by digit is more efficient than that by kana independent of the text kind.

The text that needed least keystrokes turned up to be the user e-mail text by digits, not the newspaper article used to construct the dictionary. The reason for this is that e-mail text was quite characteristic that it contains many stops (periods and commas) than other texts (see Table 6). Note that the average keystrokes that we show also measures text style as well as the efficiency of input.

The worst text by digit input was the user's book data of technical domain. For this text, DWC1 has least difference with keystrokes by KWC1-PB method. Therefore, how efficient the input by digit depends on text kind, although the fact that input by digit is the most efficient method stays with our four test data.

As a whole, for all data, DWC1 is 37.76% more efficient than KWC1-Kana, 21.82% than KWC1-PB, DP1 is 40.05% more efficient than KP1-Kana, 19.87% than KP1-PB in average.

6.5 One Stage vs. Two Stages

We also compared our method with that of T9[9] or ZI[10]. As T9 and ZI methods are word based, we also implemented a simple word based method (not using completion) DW1 and then compared it with DW2. Because our system is based on HMM, conversions of DW1 and DW2 are both estimated by HMM.

As is discussed, DW2 has two stages of selection, first to select kana and then to select the final kanji form of the word. As T9 and ZI methods both are not open to public, we guessed the minimal input method using two stages and implemented it as follows. First, user types in a word by digits of consonants, then presses a key to invoke digit to kana conversion. User selects his target then presses another key to invoke kana to kanji conversion and then selects the final form.

Because there are words of final form that only consists of kana, we made the system not to pass these words to the second selection process of kana-kanji conversion. We could sum up the number of keystrokes by addition of that needed for input, selection of kana sequence (at least 1 needed to invoke digit kana conversion for all words)[7] and selection of target (at least 1 *only* for words that contain kanji).

[7]When the target appears as the best, user can directly go onto the second selection process. Therefore the number of keystrokes needed to choose nth target will be n, containing the key stroke to *invoke* the conversion process.

Table 9: Number of keystrokes per word for DW1 and DW2

		Newspaper		Personal text	
	action	general	economics	e-mail	book
DW1	select	2.298	1.783	1.959	2.599
	total	4.729	4.649	4.285	4.845
DW2	select1	1.644	1.445	1.567	1.717
	select2	0.555	0.621	0.385	0.510
	total	4.630	4.933	4.278	4.473

The results are shown in Table 9. Here, keystrokes for input action is not indicated, because it is in common to DW1 and DW2. The result of second selection is total number of keystrokes divided by *whole* number of words (also words that does not contain any kanji.) so that values could be summed for select1 and select2.

For DW2, we could see that the number of keystrokes for selection is small at each stage. As a result, DW1 is quite defeated by DW2 for the test data of user's book. Having that the user's book was also a hard task for DP1 and DWC1 (see Table 8, first row), we should look for some other method to specialize the system to the text.

For the other three, DW1 and DW2 competes well. Having these results, we think that DW1 is better because man-machine interaction is far simpler. Also, if we compare DWC1 and DW2, then DWC1 is better (except for the user book). Therefore, we could say that DWC1 is the better choice with its simplicity of man-machine interface and also from the efficiency point of view.

7 CONCLUSION

We have discussed an alternative Japanese text input method for mobile phones. The user inputs the sequence of digits that corresponds to the sequence of consonants so that the number of key strokes decreases to at least half. The system then needs to estimate the most probable word sequence from digits. We first verified whether such input system could be put to practical use, then argued that the word frequency and part of speech tag could be the key to solving the problem. Then we implemented our system using Hidden Markov language model. With this study we verified that our system decreases more than 35% of the key strokes of the most popular text input method used on current mobile phones.

The most important future work is to examine the potential of a personalized dictionary. Mobile phones are, by nature, for personal use. Therefore if the dictionary could be personalized based on recent context or the user's own corpus, the text input would be more efficient by eliminating the uncommon word candidates. We are currently applying a statistical learning method by extending input system described in this paper.

References

[1] J. Aoe. An efficient implementation of static string pattern matching machines. In *IEEE Transactions on Software Engineering*, 1989.

[2] E. Charniak. *Statistical Language Learning.* MIT Press, 1993.

[3] NEC Co.Ltd. Japanese text input system. In *Japanese Patent No.10-124506*, 1996.

[4] Toshiba Co.Ltd. Japanese text input system. In *Japanese Patent No.57-185528*, 1982.

[5] T. Masui. PO-BOX an efficient text input method for handheld and ubiquitous computers. In *the ACM Symposium on User Interfface Software and Technology*, pages pp.113–119, 1999.

[6] Y. Matsumoto and et. al. Manual of Japanese morphological analyzer *chasen*, 1997. Naist Technical Report.

[7] M. Nagata. *Research on Japanese with Stochastic Models.* ph.D. thesis, 1998.

[8] ASCII-24 (online news service). Tegic9 announces japanese input software for mobile phones, 2000. `http://www.ascii24.com/24/news/tech/article/2000/12/26/621457-000.html`.

[9] Tegic9. Tegic9 home page, 2000. Available from `http://www.t9.com`.

[10] ZI-Corp. Zi home page, 2000. Available from `http://207.229.18.241/`.