Optimizing predictive text entry for short message service on mobile phones[1]

Yijue How

School of Computing
National University of Singapore
3 Science Drive 2, Singapore 117543
howyijue@gmail.com

Min-Yen Kan

School of Computing
National University of Singapore
3 Science Drive 2, Singapore 117543
kanmy@comp.nus.edu.sg

Abstract

Mobile phone based SMS messaging is a ubiquitous form of communication in the modern world. However, the 12-key keypad found on many mobile phones today poses problems for text entry. As three or four letters share the same key, some form of disambiguation is required to determine which letter is intended by the user. The predictive text entry method is the most common text entry technique used in present day mobile phones. To measure the efficiency of text entry, we perform a task analysis to model the actions of users and develop keystroke-level and time-level models. A corpus of SMS messages is collected and users were video-taped to obtain timing information for data analysis. Based on these models, we propose methods to improve the predictive text entry method, focusing on remapping letters to keys and examining predictive word completion. Using the time model, we show how both techniques can be reinforce each other to yield a reduction of over 25% of time for message input over the standard predictive text entry model.

1 Introduction

Short Message Service (SMS) allows people to send or receive text messages of up to 160 characters from mobile phones. As mobile phones are an indispensable and ubiquitous tool of the modern citizen, the number of SMS messages sent has increased from approximately 4 billion in Jan 2000 to 24 billion in May 2002 (Netsize, 2003). Thus, it is essential to investigate and find ways to improve the efficiency of the text entry methods available. However, the standard 12-key keypad present on many phones poses a problem for text entry, as three or four letters share the same key. A common approach to SMS text entry is the predictive text entry method. It is a text entry technique to help disambiguate which letter is intended by the user.

In this paper, we propose methods to improve the efficiency of predictive text entry by studying the statistical properties of short message service (SMS) messages in English. We use a corpus-based approach to remap the letters on the keypad to reduce the number of keystrokes. We also examine a language model approach to complete the word that the user is currently typing. Both of these approaches were studied in the context of a publicly-available corpus of over ten thousand English SMS messages collected by the authors.

To properly assess the efficiency of SMS input, we have developed two models to measure input efficiency. We first adapt the Keystroke Level Model (KLM) from previous work in usability research for use in the SMS input domain. We then extend the model by associating keystrokes with different semantic tasks (e.g., letter entry and symbol entry). These operations are given approximate times by an analysis of videotaped sessions of users performing SMS input. The resulting operation-level model allows us to better approximate the time cost to input an SMS text message over earlier, simplistic keystroke counting model. By using the operation level model, we show that key remapping combined with input text prediction can save over 25% of the estimated time needed to input messages using the standard predictive text entry method.

After a brief discussion of text entry methods and previous work in optimizing SMS inputs, we examine the corpus collection procedure in Section 3. The keystroke and time level are discussion in Section 4. We then present our methods for optimizing input in Section 5: first, letter remapping on the keypad by using genetic algorithms, followed predictive word modeling. We conclude with a discussion of performance analysis and directions for future work.

[1] Please direct all correspondence to the second author, Min-Yen Kan.

2 Background

There are currently two main methods that are usually used on mobile phones for text entry. They are the multi-tap method and the predictive text entry method. In the multi-tap method, a user taps the key that contains the letter repeatedly until the desired letter appears. The number of taps required depends on the position of the letter on the key. For example, a user can tap the 2-key once to get 'a', twice to get 'b' and thrice to get 'c', as shown on the standard keypad in Figure 1.

Figure 1: Standard ISO mobile phone keypad. Alphabetic letters are mapped to keys '2' through '9'.

While the multi-tap method remains popular with some users, we focus on a more advanced model of input, which is called predictive text input, which is featured on many phones as the default method of input. In this method (*e.g.*, Tegic's T9 and Zi Corp's eZiText), the user presses the key that corresponds to each letter of a word once. The system uses a dictionary of words to determine which of the words the key sequence matches. Like the multi-tap method, this entry method also suffers from problems when a key sequence is ambiguous. For example, '63' on a standard ISO keypad corresponds to "mno" and "def", in which the English words "of" and "me" can be spelled. When multiple words share the same key sequence, users have to press a 'Next' key (usually '*') to move to among the alternatives. Most dictionary models attempt to order these sets of ambiguous words by relative frequency of words. Thus, if a user presses the sequence '63', "of" might be indicated as the default word as this is a more frequently-occurring word than "me". In this paper, we use the term "standard dictionary" to refer to Tegic Corp.'s T9 dictionary, and the "standard keypad" as the mapping of alphabetic keys to the 12 number keys shown in Figure 1, as standardized by the ISO (ISO, 1994).

3 Short message service (SMS) corpus for comparative research

As SMS messages differ greatly from standard written English (exhibiting shortenings, emoticons, etc.), it is vital to perform optimization of input with respect to actual messages. For example, "of" might be a more frequent word in English language texts, but since SMS messages are largely interpersonal communications, an SMS corpus might show that "me" (typed on the ISO keypad with the same key sequence) is more common. Any corpus-based method for optimizing input should therefore be done over an actual SMS collection.

We thus collected a corpus of messages as part of the basic groundwork for our research. The version of the corpus used in this study consists of over 10,000 SMS messages, and largely originate from students attending the university. Contributors were aware that their messages were to be made publicly available. To our knowledge, this resource represents the largest corpus of public-domain SMS messages for use in research by two magnitudes, the second largest being a message collection at HKU[2] with about 500 messages in mixed English and Chinese. We encourage other text entry researchers to use this corpus as a benchmark for comparative research and evaluation, which can be freely accessed via the World Wide Web[3].

The corpus of SMS messages were collected from three different sources. In order for the corpus to have sufficient depth per user, messages were first collected from a small pool of 20 selected phone users on a regular basis. The age of these users fall between 18 and 22 and they contributed 6,167 messages altogether, about 60% of the messages in the corpus. Another 602 messages are collected from the Yahoo SMS chat website[4], which shows the live SMS chat transcript of certain SMS chat rooms. The final group of messages was collected from a larger pool of users of undergraduate students. This portion of the corpus focused on collecting a variety of messages over a larger user base. Students were asked to send in up to 75 messages into a website collection program and received a small amount of compensation for their efforts. Instructions were given to submit only conversational English

[2] http://www.hku.hk/linguist/research/bodomo/MPC/SMS_untagged.pdf. Accessed February 2005.
[3] http://www.comp.nus.edu.sg/~rpnlpir/downloads/corpora/sms/. Released April 2004. Accessed February 2005.
[4] http://sg.mobile.yahoo.com/sms/chat/ychat_instructions.html. Accessed April 2004.

messages that they have sent or received from their mobile phones. They were also asked to refrain from typing in any repetitive messages. After filtering out the obvious noisy inputs, a total of 3,348 genuine SMS messages were collected from 146 individuals. This wide sample based enhances the breadth of the corpus, which helps to ensure that results derived from the model reflect a diverse population of users.

Figure 2 shows the distribution of SMS messages collected in the corpus. Users P1 to P5 represent the specific phone users whom we collected SMS messages from for the in-depth sub-collection. We were unable to identify the remaining phone users that we collected the messages from and so we are unable to show statistics of their contributions in the figure. Users W1 to W45 represent those users that have contributed more than 25 messages at the website in the breadth portion of the collection.

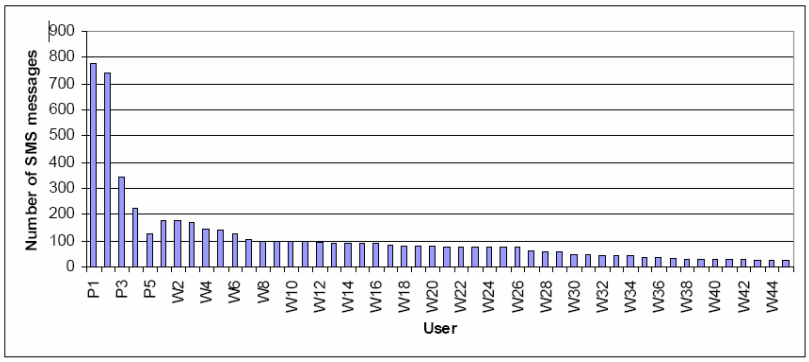

Figure 2: Distribution of SMS messages. Users who contributed less than 25 messages are not shown.

4 A operation level model for SMS entry

Previous work in benchmarking the efficiency of text entry methods have concentrated on counting the number of keystrokes by adapting the Keystroke Level Model (KLM) developed by Card *et al.* (1983). To predict the execution time of a task, a list of keystroke actions performed by the user is specified and the times required by each action are summed together. There have been several past works that adapt the KLM to predict multi-tap and predictive text entry times on mobile phones (e.g., Dunlop and Crossan, 2000; Kieras, 2001). In particular, Dunlop and Crossan (2000) gave estimates of average time by modelling interactions using the operations K, H and M (for Key press, Homing time, and Mental preparation time). In their equations, they made use of the estimated times given by the KLM for each operation. But the time stipulated by KLM was estimated for computer users using the keyboard. As the buttons on mobile phones are much smaller than those on keyboard and mobile phone users normally use one finger for input, the timings for pressing a button on mobile phone is unlikely to be similar to that of a full-sized computer keypad. Furthermore, their model does not account for punctuation mark and symbols.

We adapted KLM model by constructing an approximate time-level model, which models the fact that certain keystrokes take a longer time to input than others. (Pavlovych and Stuerzlinger, 2004) described a model that partially accounts the cognitive actions that users perform mobile text entry. We present a complementary approach to this goal by classifying each keystroke as one of 13 different operations. This operation-level model (OLM) is shown in Table 3 and attempts to group keystrokes together that are likely to incur similar time cost.

Note that the OLM described here partially accounts for spatial layout differences that are often accounted for by Fitts' Law (Fitts, 1954) and the standard KLM model. Fitts' Law models the difficulty of moving and seeking from one key to another based on the intervening distance and the target key size. We feel that our Boolean model of moving or repeating a key press is sufficient for time modelling and that Fitts' Law may not model small, mobile interfaces well. Furthermore, previous adaptations of the KLM have a single, unvarying mental preparation time which we feel does not model the differences between cognitively distinct tasks.

Using the OLM, an arbitrary SMS can then be decomposed into an inventory of its operations. Similar to previous work, we assume that the user makes no mistakes in entering the text, and follows the most straightforward manner (by choosing the method that minimizes keystrokes or time) in entering the SMS message. We explain this process using a sample artificial message "Reach home @ ard 930", shown step by step in Table 2.

In predictive text entry, only one key press is needed per character. Each key press is counted as an MPAlphaK operation, except when the current key press shares the same key as the previous one, in which case it is counted as an RPAlphaK operation.

Table 1: OLM operations and descriptions. Timing (column 3) introduced and discussed in Section 4.2.

Operation	Description	Timing in secs. (Novice / Expert)
1. MPAlphaK Move and Press Alphabet/Symbol/Space Key	The action of moving and pressing a key that corresponds to a letter in the alphabet, a symbol or the SPACE character.	2.17 / 0.70
2. RPAlphaK Repeat Press Alphabet/Symbol/Space Key	A repeat press of a key that corresponds to a letter, symbol or space when the user's finger is already on the key.	1.21 / 0.63
3. MPNextK Move and Press 'Next' Key	The action of moving and pressing the 'Next' key.	2.15 / 0.91
4. RPNextK Repeat and Press 'Next' Key	A repeated press of the 'Next' key.	
5. MPHAlphaK Move and Press and Hold Alphabet/Symbol/Space Key	Same as **MPAlphaK** except that the user holds on to the key until the character appears on the screen.	2.27 / 1.92
6. RPHAlphaK Repeat Press and Hold Alphabet/Symbol/Space Key	Same as **RPAlphaK** except that user holds on to the key until the character appears.	
7. MPDirK Move and Press Directional Key	The initial action of moving and pressing a directional key to move the cursor in the symbol table.	0.31 / 0.85
8. RPDirK Repeat Press Directional Key	Same as **MPDirK**, but for multiple repetitive presses.	
9. MPSymTabK Move and Press Symbol Table Key	This operation describes all actions that need to be performed to reach the symbol table. On some phones, this is a single '#' key press.	3.54 / 0.84
10. MPSelectK Move and Press Select Key	The action of moving and pressing a key that confirms the user's selection. This action is performed for some phone when selecting a symbol to insert from the symbol table.	4.14 / 3.16
11. MPModeK Move and Press Mode Key	This operation describes the action of moving and pressing the key that changes the case. During text entry, users can change between upper-case, lower-case and automatic-case. Automatic-case is where the first letter of each sentence will be automatically capitalized.	0.64 / 1.55
12. Wait Timeout kill (for multi-tap)	When two letters sharing the same key are entered consecutively, user need to wait for timeout before entering the second letter and after entering the first letter. Used when a user switches to multi-tap mode (e.g., when spelling a word not in the phone's dictionary)	2.16 / 1.35
13. InsertWord Inserting or spelling a word	Encompasses all actions that are performed by the user in order to insert a word into the dictionary. Does not include keystrokes that actually type the sequence of keys to spell the word.	3.58 / 3.58

Thus, to enter the word "Reach" (shown as the first word in sample message in Table 2), 4 MPAlphaK operations and 1 RPAlphaK (for the 'c', as 'a' and 'c' are on the same key) operations are necessary for the standard ISO keypad. The others operations MPHAlphaK, RPHAlphak, MPSymTabK, MPSelectK, MPDirK, RPDirK and MPModeK are calculated in a similar way. The operations MPNextK and RPNextK describe the action of scrolling through the word options by user. For example, the word "home" is the second word in the word list for the sequence '4663' on the standard keypad ("good" is the first word in the T9 model). As such typing "home" requires a single MPNextK key press after the four MPAlphaK and RPAlphaK operations needed to type the sequence, as

shown in step 3. The token "ard" is not present in the standard T9 dictionary and needs to be inserted. Therefore, the user will have to perform one InsertWord action and then spell the word out using the multi-tap input method, which results in the 5 MPAlphaK and RPAlphaK operations as shown in step 7.

Table 2: Operators per token in typing the sample message "Reach home @ ard 930".

#	Token	Actions
1	Reach	4 MPAlphaK, 1 RPAlphaK
2	SPACE	1 MPAlphaK
3	home	3 MPAlphaK, 1 RPAlphaK, 1 MP'Next'K
4	SPACE	1 MPAlphaK
5	@	1 MPSymTabK, 1 MPDirK, 1 MPSelectK
6	SPACE	1 MPAlphaK
7	Ard	1 InsertWord, 3 MPAlphaK, 2 RPAlphaK
8	SPACE	1 MPAlphaK
9	9	1 MPHAlphaK
10	3	1 MPHAlphaK
11	0	1 MPHAlphaK

4.1 Measuring Efficiency

The keystroke model is a simple model used in previous work (MacKenzie, 2002) to approximate text entry efficiency. This model measures efficiency of the text input interface by simply counting keystrokes. The sample message "Reach home @ ard 930" requires 5+1+5+1+3+1+6+1+1+1+1 = 26 keystrokes in our processing model. The efficiency of any input method then found by calculating the average number of keystrokes per message. An efficiency input rate can then be calculated by estimating keystrokes per time unit, yielding standard metrics such as characters per second or words per minute.

Although the average number of keystrokes is indicative of the efficiency of a text entry method, its main problem is that it assumes that every operation incurs the same time cost. Previous work by Mackenzie (2002) showed an inverse correlation between keystrokes per character and text entry throughput. In the process above, keystrokes were assigned to specific classes of operations. To assure that this information is properly accounted for, we devise a model in which each operation type is assigned a different amount of time. Therefore, the time required to type a message, *m*, is calculated by: $totalTime(m) = \sum_{o \in Actions} num(o) \times time(o)$, where *num(o)* is the number of times operation *o* is performed, *time(o)* is the time needed to perform *o*.

4.2 Acquiring Timing Models from Human Subjects

A critical part of the time-level model is *time(o)*, which needs to be determined. We estimate these by analyzing videotapings of volunteers whom were asked to input sample SMS messages. We used 5 actual messages from the collected corpus described in Section 3, selecting messages that would display the full breadth of operations. We limited our taping sessions to five messages to keep each session under five minutes. The selected messages are shown in Figure 3. Instructions were given to type the messages using one hand and to type the messages exactly as seen on the paper. A tutorial on how to enter input text on mobile phones was also given to participants who had no experience with SMS messaging.

1. U still wan me 2 reg e gown 4 u? But need ur add, IC n matric. Then e 3 measurement.
2. Sim lim square, shop around , maybe u chk out IT mart @ #03, in front of bullet lift, look 4 jack,
3. Matric pack, ccas & laptops
4. Hey you on your way? I cant go in yet. =(
5. Hey i'm going to be pretty late...

Figure 3: SMS messages used for acquiring timing models.

As times for individual actions are hard to determine precisely (most actions take less than 2 seconds) and still vary substantially, we time the input of an entire SMS message per subject and interpolate times for each operation type, which we feel is more accurate. Using the process described earlier, we associate a message's inventory of operations with its time cost to create a set of linear equations. Linear equations which share common variables are grouped together so that they can be solved for those variables. Because the linear equations are obtained from experimental data, no exact solution exists and only an approximate solution can be found. We find the best fit values for the variables by minimizing the sum of squared error through standard regression analysis.

A total of five subjects were taped, two of which had no previous experience with predictive text entry ("novices") and three of which had substantial experience ("experts"). We created separate models for each group. The final timing model is shown in Table 3 in the third column. Our model validates earlier claims concerning the difference between expert and novice classes of users (e.g., James and Reischel, 2001) and also recovers a shorter time for repeat key presses. This difference is less notable in expert users, who are familiar with the keypad and do not need much cognitive effort to locate the appropriate new key (about 10% more costly). On the other hand, novice users required more time and finding a new key was markedly more difficult (about 79% more costly).

Due to the small number of subjects taped, the derived model timings are stable only for operations that occurred frequently in the input messages. As such, some of the operations were conflated together to reach statistical significance. We hope to extend this work further by taping more subjects so appropriate values for all operations can be established with statistical significance.

4.3 Baseline performance of the Predictive Text Entry Method

In order to compute the number of operations required for each word in each message in the standard predictive text entry method, we need associate each word with a number of keystrokes. A critical component for words that share the same key sequence is the number of 'Next' key presses that are necessary to reach the word. For example, the key sequence '6334' spells "good" with 0 'Next' key presses and "home" with 1 'Next' key press. For each unique word in our SMS corpus, we compiled how many 'Next' key presses were necessary to find the word in question. Words not present in the standard dictionary were noted as such. Such words are assumed to be entered by the user by typing the word in multi-tap input mode and then inserting the word into the text. Auto-capitalization support (where the first letter of a sentence is automatically capitalized) was assumed.

With the above information we can predict the performance of the standard predictive text entry model. Using the keystroke model, messages in the corpus took an average of 74.004 keystrokes to input. A similar calculating was done with the multi-tap input system, yielding an average of 118.925 keystrokes. Our results show that predictive text input performs about one-third better than multi-tap by keystroke count. Using the time-level model, the predictive and multi-tap methods take an average of 59.7 and 79.3 seconds, respectively. These results support earlier findings that multi-tap is indeed slower that predictive text (MacKenzie, 2002); but differ greatly in magnitude of the results; we feel this is likely due to the different corpora used. The advantage of our study is that we use a large corpus of actual SMS messages, but a disadvantage is that the time per operation estimates are derived from a very small number of users.

5 Optimizing Text Entry

To improve upon the predictive text entry we examine two approaches: 1) remapping the alphabet to the keypad to reduce ambiguity and 2) predictive word completion to reduce keystrokes required.

5.1 Key Remapping using Genetic Algorithms

By remapping the alphabetic characters on the keypad, we can attempt to reduce the amount of ambiguous key sequences. For example, defining "6" as "mn" and moving "o" to the "7" key will disambiguate "of" and "me", but may introduce other ambiguities. To try all permutations of keypads is infeasible, as there are ${}^{26}C_3 \times {}^{22}C_3 \times {}^{20}C_3 \times {}^{17}C_3 \times {}^{14}C_3 \times {}^{11}C_4 \times {}^{7}C_3 \times {}^{4}C_4 \approx 1.5 \times 10^{19}$ possible combinations, if we keep the number of letters allocated to each key fixed. A guided search is necessary. To find a solution, we utilize genetic algorithms. We define a mapping of the 26 letters to the 8 keys used for these letters as having a higher fitness, if and only if it lowers the average time-level model estimate for the messages in the corpus. By creating many permutations the standard keypad, we can evaluate whether any improve performance. Keypads that show a performance improvement are then used in subsequent iterations to search for even better keypads.

Using this model, we swap positions of letters on the keypad. This is done by randomly picking a subset of 2 or 4 letters and swapping their positions. A swap4 mutation is illustrated in Figure 4. A position of a letter is encoded just by the number key it is assigned to, as predictive text input treats the letters on each key as an unordered set. In our OLM, letter swapping can change the number of 'Next' key presses needed to type words (i.e., MPNextK and RPNextK). The swap can also switch a keystroke from a move + press to a repeat press (i.e., from MPAlphaK to RPAlphaK). Other operation counts are unaffected by the keypad mapping. As such, a good fitness function measures the average time cost with respect to these four operations.

Figure 4: A (l) parent keypad, and derived (r) child keypad after a swap4 mutation. Swapped letters are highlighted. The child keypad is in the equivalence set of optimal keypads for our paper.

Note that the number of 'Next' key presses for each word in the corpus changes with every keypad variant. As letters have shifted, different key sequences will lead to a change in the ambiguous sequences. As such the standard number of 'Next' key presses that was derived from the standard dictionary cannot be applied. Instead, we use the frequency of the tokens in the SMS corpus to assign the ordering among ambiguous sequences: more frequent tokens are ordered before less frequent ones.

Using this asexual genetic algorithm, we produce 100 mutated keypads per iteration, keeping the 10 that have the least time cost. After 131 iterations, no changes in the ten top keypads were observed. Figure 4(r) shows one of the ten resulting keypads. The average time cost was reduced by 21% in the expert model and by 13% in the novice model. The difference is likely attributed to the large difference in cost ratio of the MPAlphaK and RPAlphaK operations in the two models: the reduction in the number of 'Next' key presses is partially negated by needing to move to another key in the novice model.

In a complementary study, Dinesh Tulsidas (2002) also examined remapping of keys for predictive input text. His work examined spatial layout optimization in conjunction with ambiguity reduction. The spatial component tries to optimize the mapping to lessen movement time between keys as governed by Fitts' law. While we do not have a spatial component in our model, our OLM accounts for these differences as move or repeated key presses as previously stated. Also, whereas our work computes the actual operations needed to input a SMS message, Tulsidas' model uses an approximation based on word length.

5.2 Predictive word completion

Many mobile input interfaces also include a word completion system, in which a partially completed word can be completed to the full word intended by the user. ZiCorp is one such corporation that uses word completion in its input method, eZiText[5]. While most current generation systems that do word completion predict words based only on the current sequence typed so far, mobile devices have increasingly larger capacity to hold language models. As such we examine a next generation approach that examines current word prediction that uses the previous word that was keyed by the user, in addition to the current key sequence typed so far.

To motivate this approach, consider a user writing the message "Meet at home later". Assuming the standard ISO keypad, a user who has typed the first two words and is keying the sequence for "home" ('4663') in will be shown "i", "in", "inn", "good" as the keys are typed, with "home" finally entered after a 'Next' key press. However, a corpus analysis can reveal that "home" is the most likely word being typed, even when only three letters have been typed, based on the previous word, "at".

In language modeling terms, we combine the letter n-gram model with a bigram word model to make a prediction. That is, we select the word with the highest conditional probability given the joint evidence from the typed sequence and previous word:

$$P(curWord \mid prevWord, keySeq) = \frac{C(prevWord, curWord)}{\sum_{w \in words(keyseq)} C(prevWord, w)} \qquad \text{(Equation 1)}$$

[5] http://www.zicorp.com/ezitext.htm, retrieved February 2005.

where *prevWord, curWord* and *keyseq* are the previous word entered, current word being entered, and the partial key sequence that has been entered thus far. *words(k)* is a function that returns the set of words which have key sequences starting with *k* and *C(x,y)* is the count of the word bigram *(x,y)* in the corpus. Counts are again estimated from the collected SMS corpus. All word completion choices can thus be ordered by their probability.

As word completion can take up a sizeable area of a phone user interface, we limit when word completion suggestions appear to the user and only select a single candidate to suggest, different from other commercial implementations. Four conditions govern whether suggestions are shown, to limit suggestion to high probability candidates:

1. Given a previous word, *prevWord*, and a key sequence, *keySeq*, a word, *curWord*, is predicted only when: (i) *P(curWord | prevWord, keySeq)*, calculated by Equation 1, is greater than the threshold of 0.5; and (ii) *C(prevWord,curWord)* must be greater than 2.
2. Given a previous word and a set of words, *W*, which share the same partial key sequence, *keyseq*, and for all $w \in W$ in which condition 1 holds, then the word that is predicted is given by:
$$curWord = argmax\ P(w \mid prevWord, keqSeq) \quad \text{(Equation 2)}$$
If more than one $w \in W$ such that *P(w | prevWord, keySeq)* is maximized, then the *w* with the longest word length is predicted.
3. If a given key sequence, *k*, codes for at least one complete word then all possible completions of *keyseq* are suggested according to their probability as calculated by Equation 1.
4. Word completion is only can only be active after the first letter of a word has been entered. That is, when *keyseq* is non-null. This is done so as to make the prediction more accurate, because one key press greatly narrows down the possible completions.

By following these conditions we can predict when word completion will suggest a completion during the processing of the OLM on a message, using the standard ISO keypad. To accept the suggestion, the user can press '0' for a space or '1' for punctuation, which completes the word and types the appropriate space or punctuation. Consider the example of a user who wants to type in "meet at home later" and who already has keyed in "meet at ". If keying the sequence '64' leads the sequence completion "home" to satisfy the conditions above, the user types '0' to key in a space and accept the word completion, saving a single keystroke. This keystroke combines the semantics of both an AlphaK and NextK keystroke, and ideally would be modeled as a new operation in the OLM. As it is not possible estimate this operation without a working system, we model this keystroke as the MPNextK operation which implicitly includes verification (mental preparation) time.

Assuming that the word completion candidate is always chosen when it is the desired candidate in message, we can use the OLM to derive the number of operations needed to code an arbitrary message, in which the number of operations for key sequence entry and 'Next' key presses are reduced. It is then straightforward to calculate the average time cost per message in the corpus, resulting in a time savings of 18% and 13% for the expert and novice models, respectively.

The improvement due to completion is slightly less than what was achieved from key remapping. Additionally, the figure is smaller than previous reports on word completion in other mobile applications. For example, PDA input of Japanese using word completion was shown to increase input rate over 60% (Masui, 1998). This discrepancy is largely due to the fact that SMS messages contain many words that are shortenings that do not benefit significantly from completion. This highlights the need to do testing on actual SMS corpora.

Can the two approaches of key remapping and word completion be combined to achieve additional cost savings? The answer is yes: we can start from one of the final keypad configurations from the remapping process and then calculate what additional savings occur when completion suggests the correct word. This is achieved by re-computing the probability function given in Equation 1. Only the *keyseq* variable has been changed, so the denominator needs to be recomputed; the bigram word model in the numerator is the same. Combining the two techniques together further reduces the cost in both keystroke and time level models to 21.8% and 28.6% from the baseline, respectively. The result shows a further reduction of around 7% over the using word completion or remapping alone.

6 Discussion

Table 3 summarizes our findings, showing the average number of seconds per message using the OLM to derive average input times for each of the input methods described in Section 5.

Table 3: Average seconds per message by entry methods, from the time-level model. Auto-capitalization included.

	Average			Expert Model			Novice Model		
Entry Method	Avg.	Std. Dev.	Improvement	Avg.	Std. Dev.	Improvement	Avg.	Std. Dev.	Improvement
Baseline	74.00	51.1	-	59.79	42.1	-	149.56	103.1	-
Remapping Letters	62.35	43.2	15.7%	47.17	33.3	21.1%	128.77	89.7	13.9%
Word Completion	63.54	46.5	14.1%	48.74	36.5	18.5%	129.37	94.3	13.5%
Combination	57.87	42.4	21.8%	42.68	30.9	28.6%	120.24	86.4	19.6%

We state the average, standard deviation and the percentage improvement of the text entry method over the baseline of each of the different text entry techniques. Results are given for a combined model as well as for separate novice and expert user models. The time model for the expert user reduces the average time required to type a message by 21.1% for the remapped keypad and 18.5% for predictive word completion. A combination of both techniques further reduces the average time, with a 28.6% improvement over the baseline. Using the combination of both techniques, an expert user would on average use 30.9 seconds to type a message and a novice user, 86.4 seconds.

The remapped keypad generally performs better than predictive word completion. This may be because we adopt a conservative strategy for completion, as we only suggest the most probable word and we start predicting only after the first key has been entered. From the tables, we can also see that the gains from these two techniques overlap as the resulting gain from the combination of both is less than the total gains from both techniques. This is likely because both techniques reduce the same type of operations ('Next' key presses).

6.1 On cognitive load

The results given in these tables do not take into account the increase in cognitive load as a result of using these improvement techniques. We are aware of this limitation, which has been a problem largely undealt with in previous work as well. Researchers at York University have recently adapted a KLM to account for verification times (Pavlovych and Stuerzlinger, 2004), which model the cognitive load a user requires to verify that a correct word or letter has been typed. This is a promising approach which can be incorporated in our model as an additional operation type.

We conjecture how cognitive load will affect the performance figures in Table 3. Actual performance gains would likely be less than reported in the table, as the cognitive load for using a remapped keypad and word completion are higher than in the standard predictive text model. More specifically, for the remapped keypad, the novice model times for the MPAlphaK and RPAlphaK operations are likely to increase, as the user has to visually scan the entire keypad for the next letter. Expert user times will not be affected much; as such users can easily recall the remapping through repeated use and will be able to touch type without needing to scan the keyboard.

The impact of cognitive load on word completion is different. The word completion model outlined here is based on the current letters of the word being typed as well as the previous word. The fact that the model uses two sources of prior evidence makes it more difficult for an expert user to predict when a completion will be suggested. Also, expert users check the screen to verify their input less often than their novice counterparts and are more likely to miss a suggestion. As such, we hypothesize that experts are less likely to benefit from word completion and will ignore the suggestion capability for many words. On the other hand, as novices verify their input often, they are more likely to benefit from this technique. We can mitigate this seeming "non-determinism" of the prediction by removing the bigram word model from the prior evidence, but this will lower the predictive power of the model.

Table 4 summarizes our hypotheses with respect to cognitive load.

Table 4: Hypothesized impact of cognitive load on the time savings for the input methods studied.

Input method	Expert Model	Novice Model
Key Remapping	Similar to model	Significantly reduced
Word Completion	Significantly reduced	Similar to model
Combined	Reduced	Reduced

7 Conclusion

We have presented two methods to improve predictive text input for mobile input devices: by remapping the letters to different keys and by predictive word completion. We predict the two methods' utility in reducing input time necessary for keying messages, which show that both methods enhance performance separately by about 15%. When the two techniques are applied together, an average decrease of 20% in necessary input time was observed.

In order to compute these figures, we developed an operation-level model (OLM) which decomposes the input of an SMS message into an inventory of operations. Operations group keystrokes that are likely similar in their execution time. Similar to the KLM for computer users, the OLM describes the different types of actions that users carry out when doing mobile text entry. When the OLM is suitably paired with timing estimates for each operator, an input time can be predicted for an arbitrary message. We estimated OLM operation times from videotape analysis of expert and novice users of predictive text entry and compute hypothesized average input times for a large (over 10K messages), publicly-available SMS corpus which we have collected. A key distinction in our work is the application of statistical methods on this corpus of actual SMS messages rather than on other full text corpora.

At present, the modelling assumes that users do not make any mistakes during input. In future work, we plan to extend this work to model mistakes and cognitive load of users through usability testing, following (James and Reischel, 2001). We also plan to refine the time estimates for operations in the proposed OLM by observing additional subjects as they key in input messages. We can then check whether the hypothesized times correlate well with actual times.

8 References

Card, S.K., Moran, T.P. and Newell, A. (1983). The Psyschology of Human-Computer Interaction, Lawrence Erlbaum, Hillsdale, New Jersey, USA.

Dunlop, M.D. and Crossan, A. (2000). Predictive Text Entry methods for mobile phones. *Personal Technologies*, pp. 134-143.

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

ISO/IEC 9995-8. (1994). Information systems – Keyboard layouts for text and office systems – Part 8: Allocationg of letters to keys of a numeric keypad, International Organization for Standardization.

James, C.L. and Reischel, K.M. (2001). Text input for mobile devices: Comparing model predictions to actual performance. In *Proc. Of CHI 2001*. pp. 365-371.

Kieras, D. (2001). Using the Keystroke-Level Model to Estimate Execution Times. University of Michigan.

MacKenzie, I.S. (2002). KSPC (Keystrokes per Character) as a Characteristic of Text Entry Techniques, In *Proc. Of 4th Int'l Symp. On HCI with Mobile Devices*, Springer, Germany, pp. 195-210.

Masui, T. (1998). An Efficient Text Input Method for Pen-based Computers. In *Proc. Of CHI 1998.*

Netsize (2003). European SMS guide, February 2003.

Pavlovych, A. and Stuerzlinger, W. (2004). Model for non-Expert Text Entry Speed on 12-Button Phone Keypads, In *Proc. Of CHI 2004.*

Silfverberg, M., Mackenzie, I.S. and Korhonen, P. (2000). Predicting text entry speeds on mobile phones. In *Proceedings of the ACM Conference on Human Factors in Computing Systems – CHI 2000*, New York : ACM, 2000, pp. 9-16.

Tulsidas, D. (2002). Optimal Predictive Text Layout for Mobile Phones, Master's thesis, Imperial College of Science, Technology and Medicine. September.