Model for non-Expert Text Entry Speed on 12-Button Phone Keypads

Andriy Pavlovych **Wolfgang Stuerzlinger**
Dept. of Computer Science,
York University
Toronto, Ontario, Canada
http://www.cs.yorku.ca/{~andriyp|~wolfgang}/

Abstract
In this paper we present a new model for predicting text entry speed on a 12-button mobile phone keypad. The proposed model can predict the performance of novice users. Like other models for text entry, the proposed model includes a movement component based on Fitts' law and a linguistic component based on letter digraph probabilities. It also adds cognitive delay times before key presses and takes into account the fact that Fitts' law cannot model multiple presses of the same key accurately. Finally, we compare the prediction of our model to previously published experimental results, demonstrate that it fits observed results for novices very well, and list some observations about learning.

Categories & Subject Descriptors: H.1.2 [**Models and Principles**]: User/Machine Systems – human factors.

General Terms: Human Factors, Performance, Theory, Experimentation.

Keywords: text entry, mobile phones, model.

INTRODUCTION
Most modern mobile phones include the capability to send and receive short text messages. Many people use this facility, and, in fact, during the last decade there has been phenomenal growth of the number of text messages sent. In the year 2002, more than 24 billion messages were transmitted each month through GSM networks alone [5]. This is remarkable, given that entering text on a phone keypad is not easy, for there are significantly fewer buttons on a phone keypad than there are letters in a language alphabet. Various text entry techniques for phones have been developed to deal with this issue, each trying to find the best way to use the 12 buttons on a keypad to enter text.

Designing text entry methods for computing systems is not an easy undertaking: eventually one needs to build a prototype device and to conduct user studies in order to be able to judge how the newly developed technique compares to existing ones. Consequently, a model that predicts the performance of a new method as closely as possible without the need to do prototyping would be very valuable.

Text Entry Methods for Mobile Phones
The traditional 12-button keypad (Figure 1) consists of ten number keys and two additional symbols (* and #). The letters are assigned to the keys in alphabetical order. Although there are some minor variations, most keypads follow the ITU E.161 standard, also known as ANSI T1.703-1995/1999 or ISO/IEC 9995-8:1994. One can read more on phone keypads and related standards in [15].

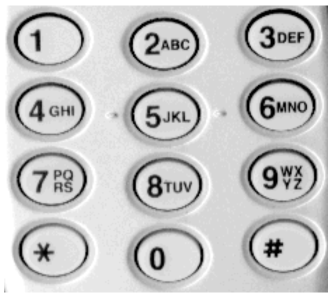

Figure 1. The standard 12-key keypad (Mitsubishi G310).

In the following subsections, we will briefly describe the existing techniques for text entry on phone keypads.

Multitap
Most phones offer *Multitap* as the standard choice for text entry. In order to enter a letter with *Multitap*, a user presses the corresponding key repeatedly until the letter appears (e.g. press '2' once to enter 'a', twice to enter 'b' and so on). A notable difficulty with *Multitap* is entering consecutive letters that appear on the same key (a problem called *segmentation*). There are two ways to deal with this situation. One alternative is to use a timeout after which the system advances to the next letter. Another alternative is to advance the cursor using a dedicated key. The first approach requires fewer keystrokes; the second tends to be faster for expert users, even though the number of key presses is greater [10].

Less-Tap

Less-tap [14] differs from *Multitap* in the order in which the letters appear upon pressing a key. The objective of this idea is to allow the entry of the most frequent letter on each button with one keystroke, the second most frequent letter with two keystrokes and so on. This reduces the average number of keystrokes by about 25%, to 1.44 key presses per character.

Two-Key

As the name implies, two key presses are required for each letter. The first press selects the group of letters (e.g. '4' selects GHI). The second press then selects the letter from the group (e.g. '2' selects H). Another interesting two-key approach, *MessagEase* is described in [13]. However, it uses a non-standard layout. In these approaches there is no issue of segmentation. Both of these two-key based methods are not much faster than *Multitap* [16] and will not be discussed further.

All of the systems mentioned so far support *eyes-free input* [12], the ability to enter text without having to visually verify the result. This, of course, assumes the user had an opportunity to memorize the layout of the letters.

Dictionary-Based Disambiguation

There are several text entry techniques that disambiguate input based on a dictionary: *T9* from Tegic Communications, *iTap* from Motorola, *eZitext* from Zi Corp. All of them act very similarly – the user presses each key only once for each entered letter. The system tries to match all possible interpretations of the entered key sequences to the words that are contained in the dictionary. The most probable word is the default choice offered. If this is not the word intended by the user, he or she has to press a special key (NEXT) until the intended word appears. This means that text entry speed goes down if a desired word is one of the less likely choices in the dictionary. This class of systems fails if the word is not in the dictionary or is misspelled. In principle, eyes-free input is impossible with these techniques.

Prefix-Based Disambiguation

The only system that uses prefix-based disambiguation is *LetterWise* [10], where the *prefix* is the sequence of letters preceding the current keystroke. It guesses the most probable next letter based on what the user entered previously. Up to three consecutive letters are used in the presented implementation. Since *LetterWise* uses a database of prefixes instead of words, the technique does not fail when a user attempts to enter non-dictionary words. To deal with the case when the letter that initially appears is not the desired one, the system employs a NEXT key, as an alternative to multiple key presses. Like dictionary-based disambiguation methods, *LetterWise* forces users to pay close attention to the screen while entering text, to verify that the prediction of the system matches the user's intention. Therefore, eyes-free input is not possible.

Methods That Use Non-Standard Keypads

The following two methods use keypads different from the standard 12-button phone keypads and are described here only to provide an overview over alternative approaches. They will not be considered further in this paper in order to concentrate on 12-button-based text entry.

While miniature *QWERTY* keyboards seem to be a natural choice for text entry, their use in mobile phones is not widespread, mainly due to their (even slightly) larger size and the inability to touch type. Also, single-handed performance on miniature *QWERTY* keyboards is significantly lower than double-handed performance. Having said that, it has been demonstrated that miniature keyboards are a viable choice for personal digital assistants (PDAs) where there is a need for faster text entry methods and the size is slightly less of a constraint. Another technique, *Fastap* [8], also uses more than 12 buttons. Each letter has a dedicated small button, and the button for space is double-sized. The buttons are arranged in a 4-by-7 grid (28 'nodes' and 18 'cells') and the letters are assigned to the buttons sequentially, in alphabetical order. To enter a letter, the user simply presses the corresponding key.

Fitts' Law

Fitts' model is an important component of our model. Fitts' law [4] is a model for serial fast, aimed movements. It is expressed as:

$$MT = a + b \cdot \log_2(A/W + 1)$$

where A is the amplitude of the movement (e.g. the distance between buttons on a keypad), and W is the width of a target (e.g. a button). The log term in the equation is called the index of difficulty (ID):

$$MT = a + b \cdot ID$$

One of Fitts' law's limitations is that it assumes that the movements are only limited by human motor abilities. We will see later that in some cases the movement time is only a small fraction of the total time needed to perform a key press. Another limitation is that Fitts' law does not work well for motions that have an ID of zero (e.g. key repeats) [18].

The coefficients a and b are usually determined empirically for a given device (e.g. tablet, keypad) and the interaction style (e.g. pointing with a stylus, pointing with a finger, pressing with a thumb). Silfverberg *et al.* [16] determined the coefficients for the phone keypad used here. The reported Fitts' law parameters were a = 176 ms and b = 64 ms/bit (Nokia 5100 series, thumb entry).

Existing Models for Text Entry

Two models for text entry on a 12-button keypad were presented previously. Both of them were designed to predict expert (or *peak*) text entry rates for various text entry methods.

Table 1. Existing model predictions and mean entry speeds observed for novices and experts (wpm); see text for details.

Method	*KLM Model [2],[3]*	*Silfverberg Model [16]*	*Novice [14]*	*Novice [7]*	*Novice newspaper [7]*	*Novice chat [7]*	*Expert [7]*	*Expert newspaper [7]*	*Expert chat [7]*
Multitap	18.35	22.3	7.15	7.98	5.59	10.37	7.93	5.33	10.53
T9	24.97	40.6		9.09	7.21	10.98	20.36	15.05	25.68
Less-Tap	23.47	26.8	7.82						

The KLM Model

The first model, a keystroke level model by Card *et al.* [2], is one of the earliest predictive models for human-computer interfaces. The model tries to account for times of individual key presses, including the "mental preparation times" before each. Dunlop and Crossan [3] used this model to evaluate their text entry technique (a predictive method similar to *T9*) alongside *Multitap*. In their paper, the obtained predictions are 14.9 and 17.6 words per minute (wpm) for *Multitap* and the predictive text entry method respectively. We recomputed the predictions of the model considering that other models assume an average word length of 5 and other factors[1]. The revised predictions appear in Table 1.

Major deficiencies of the KLM model stem from the fact that it was not intended to be an accurate and comprehensive model of *text entry* but merely as a tool to analyze human-computer interaction in general. The model does not rely on Fitts' law to determine the movement time between keys and assumes that all key presses take an equal amount of time; that is, it assumes that pressing the sequence of buttons 2-3-2 takes as much time as 2-9-2 and as much as 2-2-2. Also, the mental preparation component of the model is assumed to be the same before each key press, which may not be true for phone text entry methods that require more than one press for certain letters.

The Silfverberg Model

The second model was presented by Silfverberg *et al.* [16] and is largely based on the model of Soukoreff *et al.* for soft keyboards [17]. It contains two parts: a movement model based on Fitts' law [4] and a linguistic model to determine the distributions of the key digraphs for the given corpus. Like the previous model, this one assumes expert performance with no errors.

The Silfverberg model predicts a text entry rate of 20.8 wpm for *Multitap* (without a timeout kill button) and 40.6 wpm for *T9*. Again, for the purposes of this paper, we recomputed the numbers, to reflect the fact that in the experiments, the results of which we were using to verify the model, a timeout of 1.0 s was used, instead of 1.5 s. The revised predictions appear in Table 1.

A thorough description of the models is beyond the scope of this paper; interested readers are encouraged to refer to the original texts. Note that the two models give different predictions, especially for *T9*.

Published Experimental Results

Study of T9 vs. Multitap

An interesting result is a study that evaluates *T9* and *Multitap* in an experiment with both novice and expert users [7]. The results are summarized in Table 1. As one can see, the actual observed rates do not agree with either model. Also, in [7], the authors assert that the notion of an *expert* text message user is unrealistic due to the very nature of the task. They report that a typical SMS user sends only about 20 messages per month; which seems to be consistent with what we were able to observe. As the messages themselves are short, there would hardly be an opportunity for the users to ever become experts[2]!

Study of Multitap vs. Less-Tap

In another user study [14], the authors were able to observe improvements of their technique, *Less-Tap*, over *Multitap* ranging from 0% to around 20%, depending on the user, with the average being around 10%. While the range of improvement predicted by the models was reasonably consistent with the results that were experimentally obtained (the KLM model predicted the improvement of 28% and Silfverberg model 20%), the actual text entry speeds observed were much lower than predicted. Again, please refer to Table 1 for a comparison.

Trying to account for and to explain the observed discrepancies between the predictions and the experimental results has led us to the development of a new model.

[1] We believe that the equation for *T9* in [3] is incorrect as it includes the time to press space twice: as a part of the word's 5.98 characters and as a part of the final sequence, bringing the average number of key presses to about 7.02 per word rather than 6.02 mentioned in the paper. We corrected it by replacing the factor k_p by (*word length* – 1).

[2] It should be noted, however, that in many European countries as well as in some parts of Asia, the use of phones for text messaging is very common. In such countries there exist significant groups of users whose performance is close to expert level.

A MODEL FOR TEXT ENTRY ON 12-BUTTON KEYPADS

The model that will be described in the paper can largely be viewed as an extension of the aforementioned models of Silfverberg *et al.* and Card *et al.* However, the new model is also applicable to non-expert users. This is a very useful extension, since, as stated above, many people never reach the expert level.

Non-Motor Components in Text Entry

Obviously, humans are not just precise finite state machines that transform the intended text into a sequence of performed keystrokes at a speed limited only by the physical capabilities of their limbs. They tend to make pauses to verify the progress they have made so far, mainly to convince themselves that they haven't made an error, or to prepare for their next step. Humans also make mistakes (e.g. in the text entry task they hit a key different from the one intended). These are factors that are possible to predict and quantify based on existing statistical data and consequently can be incorporated into a model. Other factors are much harder to predict, for example, the use of different strategies to perform the same task (e.g. in our case: focusing the visual attention on the keypad, on the display, or on the environment). Such factors are not likely to be included in any model.

From analyzing the key log data of previous phone text entry experiments [14] and our own observations we identified the following significant components:

1. Re-reading the phrase to be entered (present only in text entry experiments; most people prefer not to memorize the phrase they are entering).
2. Figuring out which letter of which word has to be entered next (spelling out the word).
3. Determining on which button the next letter is located and how many key presses are needed (visual search and/or memory recall operations).
4. Determining where the button is located (in case of the 12-button phone keypads; this is usually a memory recall operation).
5. Keeping count of the number of presses made while performing repeated presses of the same key.
6. Visually verifying the result of the performed key press[es] (especially important if the technique employs some kind of non-determinism or prediction).

Most of these components can be derived from the GOMS model [2]. Of course, we cannot be absolutely certain that the identified components are present for all users at any given point in time. Yet, we find it reasonable to believe that they are likely to be present on average for most users. We will demonstrate later that one can find these components even in experienced users. The times corresponding to these components will vary depending on the level of expertise of the user and the strategy employed. Also, there will be some very common words, like "the", "in", "to", which may be entered almost routinely.

At first we considered measuring the individual components described above. However, as multiple components can easily be present between successive key presses, we chose to concentrate only on the combined times separating those presses, which are the times we can measure directly.

In the next section, we describe the experiments we used to estimate the amounts of time corresponding to the aforementioned components.

Determining Non-Motor Components – Method

In all the experiments, the participants used a Nokia 5190 telephone handset connected to a computer as an input device. The equipment was identical to the one employed in [14]. The software was extended to include additional text entry algorithms.

Experiment 1: Finding a Key and Pressing it Repeatedly

The goal of the experiment was to measure the time that precedes a key press in cases when no verification of the outcome was necessary (that is, when the text entry method was fully deterministic), as well as the time it takes to perform a repeated key press. Also, in order to test the assumption that, on a 12-buttton keypad, letters are slower to enter then numbers we had users enter both letters and numbers and then analyzed the times separately.

An example of the sequence of characters participants had to enter is "`cccc 99 zzz 5 i yyy kkkkk b 44 rrr`", meaning "press the button with 'c' four times, the button with '9' two times and so on. The entry technique automatically disambiguated the letters for display and also automatically added spaces. The spaces in this experiment served only as visual separation of the symbol groups.

The times preceding the first key press for each symbol (I_{num}, I_{ltr}), which roughly correspond to components 1 through 4 above, were computed as differences between the measured times for the first press and the times that would have been needed according to Fitts' law. The times to perform repeated key presses (R_{count}), which correspond to component 5, were determined as the differences between the timestamps of the repeated key presses and the ones of the immediately preceding key presses.

Experiment 2: Verifying a Character

The objective of this experiment was to measure the time to perform a key press in cases when some visual verification of the entered character was required. Only letters were entered, and there were no spaces. The entry method used was similar to *Multitap*, with one small difference: to guarantee that verification was always required, the order in which the letters appeared upon a key press was random each time. The participants were instructed to "press the corresponding button one or more times until the letter appeared". For example, for the phrase "`cwjqwmfzjoipxduasnewr`", they had to press '2' until 'c' appeared, then press '9' until 'w' appeared and so on.

The time to perform a repeated press while verifying the letter (R_{verify}) which corresponds to components 6 was again determined as the difference between the timestamps of the repeated key press and the preceding one.

Experiment 3: Verifying a Word

In the last experiment, we measured the time needed to verify a word and to press an appropriate button (V_w or V_{w2}), which corresponds to component 6. This kind of operation is present in systems that use word-based prediction.

The text entry method used was similar to *T9*. In order to have more cases when word verification was needed, we implemented the technique in such a way that, unlike in *T9*, the word that appeared by default was not *the most* probable but was instead any random matching fragment that existed in the dictionary. The participants were instructed "to press the corresponding button once for each letter in the word and to press the NEXT key (#) at the end of the word until the correct word appears".

The first and all subsequent presses of NEXT were analyzed separately, as the initial press also involves an inter-button movement while the others do not. The used set of phrases is described in [11].

Participants

There were 12 participants in the test, recruited through advertisements posted on the university campus. Five participants were female, one was left-handed, and three were frequent users of text messaging. Age ranged from 18 to 33 with a mean of 24.5. All had extensive computer experience (seven years or more). One did not own a cell phone. One reported using text messaging on the cell phone daily, another two used it weekly; all others used it very infrequently, if at all. All participants were compensated upon completion of the user study.

Results

Table 2 below shows the results from the experiments 1, 2, and 3 (all values are in milliseconds, standard deviation in round brackets).

Table 2. Results from Experiments 1 through 3.

Component	*Time (SD), ms*
Initial time before numbers, I_{num}	701 (383)
Initial time before letters, I_{ltr}	1285 (588)
Repeated press while counting, R_{count}	272 (83)
Repeated press while verifying, R_{verify}	411 (114)
First presses of a NEXT key, V_w	1088 (371)
Subsequent presses of a NEXT key, V_{w2}	672 (254)

The most remarkable result is that the times are much greater than one would expect by considering human motor performance alone. This discrepancy will be discussed later in the paper.

Time to Enter Character – New Movement Model

From the above experiment, we can derive the times required to enter text using different text entry systems. We chose not to consider two-key input methods and other methods, which use non-standard keypads.

Multi-press Input Methods

Multi-press input methods include *Multitap* and *Less-Tap*. For these methods, the time to enter a character can be described as:

$$T_{\text{char}} = I_{\text{ltr}} + T_{\text{Fitts}} + N \cdot (R_{\text{verify}}) + [T_{\text{timeout}}] \quad (1)$$

T_{Fitts} is the time needed to move the finger from the preceding key to the current key, as computed from Fitts' law equation. N is the number of additional presses (after the first, e.g. one for double or two for triple). T_{timeout} is the time that the user would have to wait for if the current character is located on the same button as the previous one. In our studies, we didn't use the timeout kill key, but the model can account for that, if desired (the value for T_{timeout} that we used was 1000 ms).

Predictive Input Methods

Predictive input methods include *T9*, *iTap* and others. There are several strategies possible with such methods [7], [10], [16]. However, it is often expected that users ignore the display until they finish entering the current word. Sometimes users have to press a NEXT key, which, statistically, accounts for less than 1% of the total number of presses [9], [16]. However, each time before pressing NEXT, the users would need to verify the current result, a task that takes over one second (V_w, Table 2)!

Thus, the model for the time to enter a character with a predictive method is:

$$T_{\text{char}} = I_{\text{ltr}} + T_{\text{Fitts}} + N_1 \cdot V_w + N_2 \cdot V_{w2} \quad (2)$$

N_1 is the number of first presses of the NEXT key (either present – one or absent – zero). N_2 is the number of additional presses (after the first, e.g. one for double or two for triple).

Linguistic Model

A linguistic model contains information about the frequency of different letter-pairs (digraphs). In our case, it is based on the letter-pair data from the British National corpus [1] and may be slightly different from the one used in [16]. The model is represented by a 27×27 matrix, the 27 characters being the 26 English letters and a SPACE symbol. Each cell p_{ij} in the matrix is the probability of the corresponding letter pair in the corpus, so that all cells sum to unity.

Combining the Models

Now, that we have both a movement time and a linguistic model, we combine them to obtain the model that predicts the text entry rate for a text entry system and a language:

$$T_{\text{char\_in\_corpus}} = \Sigma_i \Sigma_j \, (p_{ij} \cdot T_{\text{char ij}}) \qquad (3)$$

$T_{\text{char\_in\_corpus}}$ is the average time to enter a character in the corpus using a selected text entry method, in seconds per character. $T_{\text{char ij}}$ is the average time to enter a character j after character i. To convert this number to words per minute (using the common assumption of 5 characters per word), we use the following expression:

$$WPM = (1/T_{\text{char\_in\_corpus}}) \cdot (60/5) \qquad (4)$$

VERIFYING THE MODEL

Model Predictions for Various Text Entry Methods

At this point, we can apply all the collected data to compute predictions for text entry speeds. The obtained predictions are listed in Table 3. As mentioned previously, we used Fitts' law coefficients from [16] (the ones for the thumb) since the same telephone handset was used in all the studies. As expected, the predictions show that *Multitap* is the slowest, *Less-Tap* is a bit faster (by 11%), and *T9* is the fastest (16% faster than *Less-Tap* and 29% faster than *Multitap*).

Table 3. Model predictions and experimental results (WPM).

Technique	*KLM model*	*Silfverberg model*	*Novice, newspaper [7]*	*Results from [14]*	***New model, novice***
Multitap	18.35	22.3	5.59	7.15	**5.87**
Less-Tap	23.47	26.8		7.82	**6.53**
T9	24.97	40.6	7.21		**7.58**

Experimental Data

For comparison, all relevant data is summarized in Table 3. From [7], we took the results for the novice newspaper condition, since our linguistic model was derived from British National corpus, which is much closer to newspaper text than to "chat" text.

Even short term text entry experiments, such as [14], observe learning effects in practice, which explains the fact that our model underestimates the experimentally observed values. Probably for that very reason, the match is better for novice newspaper users in [7].

The study in [7] and [14] differ in the number of phrases entered (8 and 60 respectively). Therefore, the numbers from these two studies differ due to non-equal amounts of fatigue and learning involved. It is also possible that the small set of phrases chosen for [7] was favourable to *T9*. Yet, in our opinion, in both cases, the entry speeds observed for novices appear to be reasonably consistent with the ones predicted by the new model. In [10], the issue of text entry in predictive systems is investigated a bit further. Interested readers are encouraged to refer to the original text.

Furthermore, notice the dramatic differences in relative predictions for *T9* by different models: the *KLM* model, *Silfverberg's* model, and the new model predict differences between *T9* and *Multitap* of 36%, 82%, and 29% respectively, while the measured difference is 28%. Even though the ranking of the methods stays the same, one can easily see that using expert models for non-experts can be misleading.

Observations about Changes of Coefficients with Time

By varying the coefficients, it may be possible to adapt the model to predict the performance of users at various levels of expertise. Obviously, the coefficients of our model should decrease in magnitude with practice (see [5], for example).

Short Term Changes

Even though our experiments were short (less than 20 minutes for each component), we observed significant changes in some of the components.

The time to make the first press for a letter decreases over time. However, the time to enter a number stays practically constant or increases slightly, probably due to fatigue effects (see I_{num} and I_{ltr} in Figure 2). Due to the short test and the associated large variance, it is not possible to observe that the change for letters has the expected power-law trend [5].

The time to make an additional key press while keeping track of the number of key presses (see R_{count} in Figure 2) decreases slightly at the very beginning, and then stays approximately constant.

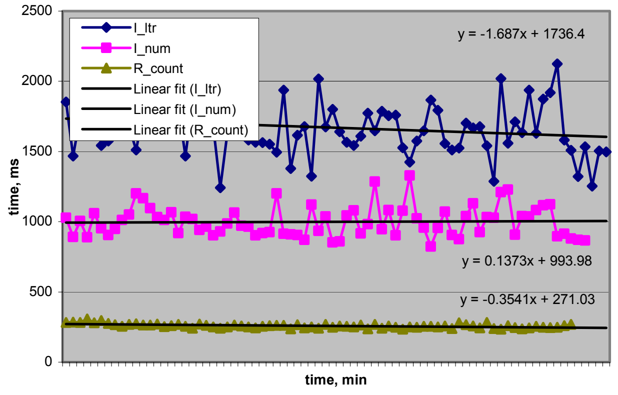

Figure 2. Changes of I_{ltr}, I_{num}, R_{count} with time.

The time to make an additional key press while verifying the entered character (R_{verify} in Figure 3) gradually decreases with time. Clearly, a long-term experiment is needed to determine the exact shape of the curve.

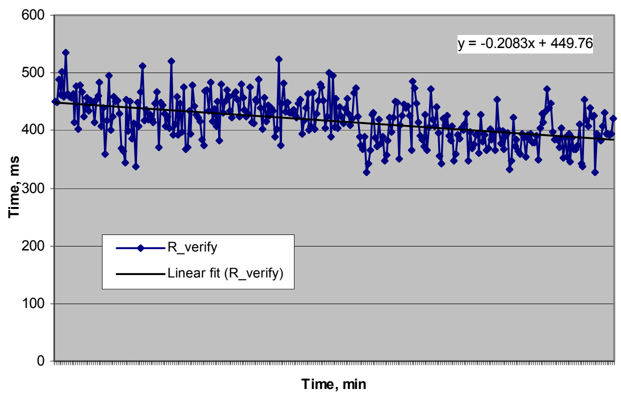

Figure 3. Change of R_{verify} with time.

The time for the first press of the NEXT key (V_{w} in Figure 4) decreases slightly at first, and then appears to level of.

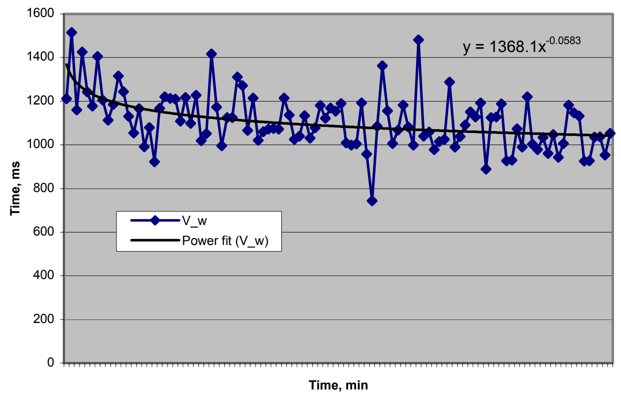

Figure 4. Change of V_{w} with time.

Long Term Changes

For this analysis, we used the results for *Multitap* of MacKenzie *et al.* [10] where they compared *Multitap* and *LetterWise* in a longitudinal study. Their setup was different from ours in that they used a large desktop-size numeric keypad. The sessions in the study were approximately 30 minutes long. Based on their data for text entry speed and equation (1) of our model, we estimated the values of the coefficients for all sessions. The following graph demonstrates our estimate of I_{ltr} and the extrapolation based on the power-law for 20 more sessions. Please note that even at the 40th session (approximately 20 hours), the cognitive delay before the first press of a letter key is still far from zero.

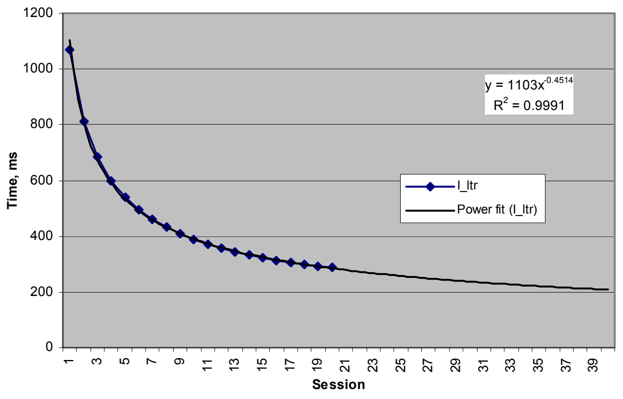

Figure 5. Coefficient I_{ltr} as a function of time.

We hypothesize from this data that I_{ltr} is the strongest factor in learning, and will account for *most* of the change in the text entry speed as users get more and more familiar with the keypad.

Please note that even though we described our observations on how the coefficients change with time, we cannot claim that we can model learning of text entry methods effectively. We consider this a subject for future work.

DISCUSSION

How to Obtain Better Predictions

Our model for novice multi-press and *T9* users assumes that the users are *complete* beginners (e.g. verification of the last character was needed every time). Just as the concept of an expert user of text entry systems is not realistic, a concept of a complete novice user is not very practical either. Usually, users learn some features of a technique very quickly. E.g. even before one uses *Multitap* for the first time one can quickly observe that the first button with letters has 'ABC' on it and utilize this knowledge. Further learning happens gradually through memorization of the layout and through becoming accustomed to the text entry method. The first manifests itself in that users remember where the most frequent letters are; the second is evident in the decrease of the coefficients as shown in the results. Also, many users acquire the skill to quickly enter very common words. In the equations above, one could model this gradual behaviour by replacing the coefficients I_{ltr}, R_{verify} by I_{num} and R_{count} for certain letters (e.g. for the most frequent letters in the alphabet or the letters that are in the first position on a key). This models that certain key sequences become well known and the user does not need to visually verify the result anymore. However, we want to emphasize that we are only speculating about these learning factors, as we do not have scientific proof yet.

SUMMARY

In this paper, we presented a model that can predict the text entry speed on standard 12-button telephone keypads for novices. The major difference between this model and existing ones is that it includes not only movement times but also the mental overhead in its predictions. Thus it can more accurately predict the average performance of non-experts. The values computed by the model (5.87 wpm for *Multitap*, 6.53 wpm for *Less-Tap*, and 7.58 wpm for *T9* for novice users) are consistent with those experimentally observed. We also made some initial observations about the process of learning in text entry methods.

ACKNOWLEDGMENTS

The authors would like to thank Scott MacKenzie and William Soukoreff for sharing their extensive knowledge on text entry methods, as well as NSERC for funding.

REFERENCES

[1] British National Corpus, available at <ftp://ftp.itri.bton.ac.uk/bnc>.

[2] Card, S.K., Moran, T.P., and Newell, A. *The Psychology of Human-Computer Interaction*, Lawrence Erlbaum, Hillsdale, NJ (1983).

[3] Dunlop, M. D., Crossan, A. Dictionary based text entry method for mobile phones, *Second Workshop on Human-Computer Interaction with Mobile Devices,* Edinburgh, Scotland (1999), 5-7.

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

[5] GSM World Association. <www.gsmworld.com>.

[6] Isokoski, P., MacKenzie, I. S. Combined Model for Text Entry Rate Development, *Extended Abstracts of the ACM Conference on Human Factors in Computing Systems – CHI 2003*, ACM, New York, NY (2003), 752-753.

[7] James, C. L., and Reischel, K. M. Text input for mobile devices: Comparing model predictions to actual performance, *Proceedings of the ACM Conference on Human Factors in Computing Systems - CHI 2001*, ACM, New York, NY (2001), 365-371.

[8] Levy, David. The Fastap Keypad and Pervasive Computing, *Proceedings of the First International Conference, Pervasive 2002*, LNCS 2414, Springer, Heidelberg, Germany (2002), 58-68.

[9] MacKenzie, I. S., KSPC (Keystrokes per Character) as a Characteristic of Text Entry Techniques, *Proceedings of the Fourth International Symposium on Human-Computer Interaction with Mobile Devices*, Springer, Heidelberg, Germany (2002), 195-210.

[10] MacKenzie, I. S., Kober, H., Smith, D., Jones, T., & Skepner, E. (2001). LetterWise: Prefix-based Disambiguation for Mobile Text Input. *Proceedings of the ACM Symposium on User Interface Software and Technology – UIST 2001*, ACM, New York, NY (2001), 111-120.

[11] MacKenzie, I. S., Soukoreff, R. W. Phrase Sets for Evaluating Text Entry Techniques, *Extended Abstracts of the ACM Conference on Human Factors in Computing Systems – CHI 2003*, ACM, New York, NY (2003), 754-755.

[12] MacKenzie, I. S., Soukoreff, R. W. Text Entry for Mobile Computing: Models and Methods, Theory and Practice, *Human-Computer Interaction,* 17, (2002), 147-198.

[13] Nesbat, S. B. Fast, A System for Fast, Full-Text Entry for Small Electronic Devices, available at <http://exideas.com/ME/ICMI2003Paper.pdf>.

[14] Pavlovych, A., Stuerzlinger, W. Less-Tap: A Fast and Easy-to-learn Text Input Technique for Phones. *Graphics Interface 2003*. Also available at <http://www.cs.yorku.ca/~wolfgang/publications.html>

[15] Phone Key Pads, <http://www.dialabc.com/motion/keypads.html>.

[16] Silfverberg, M., MacKenzie, I. S., Korhonen, P. Predicting Text Entry Speed on Mobile Phones. *Proceedings of the ACM Conference on Human Factors in Computing Systems – CHI 2000*, ACM, New York, NY (2000), 9-16.

[17] Soukoreff, R. W., MacKenzie, I. S. Theoretical Upper and Lower Bounds on Typing Speed Using a Stylus and Soft Keyboard. *Behaviour & Information Technology*, 14 (1995), 370–379.

[18] Soukoreff, R. W., MacKenzie, I. S., Using Fitts' law to Model Key Repeat Time in Text Entry Models. Poster presented at *Graphics Interface 2002*. Also available at <http://www.cs.yorku.ca/~will/>.