

TEXT ENTRY ON 12-BUTTON KEYPADS: TECHNIQUES AND MODELS

A Thesis

Presented to

The Faculty of Graduate Studies

of

York University

by

ANDRIY PAVLOVYCH

In partial fulfillment of requirements for the degree of

Master of Science

May 2003

© A. Pavlovych, 2003

## Abstract

The ability to enter text on phones is crucial for utilizing the Short Message Service (SMS) of digital cellular networks. Standard 12-Button phone keypads are not naturally suited for text entry as each button on them encodes 3 or 4 letters. This thesis presents a new technique, *Less-Tap*, to enter text on a mobile phone keypad. The new method is similar to the traditional *Multitap* method in that it requires the user to press buttons repeatedly to get a required letter. However, in *Less-Tap*, the letters are rearranged within each button according to their frequency in the language. This way, the most common letters require only one key press. A user study was conducted to compare *Less-Tap* against *Multitap*. It shows that the proposed method is significantly faster than the traditional one.

The thesis also presents a new model for predicting text entry speed on 12-button mobile phone keypads. The model is of special interest as it 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 adds a mental preparation time before key presses and considers the fact that multiple presses of the same key cannot be modelled accurately by Fitts' law. Finally, the prediction of the model is compared to previously published experimental results.

### Acknowledgements

I would like to thank my supervisor, Dr. Wolfgang Stuerzlinger, for all his help during the course of my studies at York University.

Also, I wish to thank Scott MacKenzie for providing the software for the experiments and William Soukoreff for his valuable insights on how to conduct user studies and to analyse text entry techniques, as well as NSERC for funding of the research project.

## Table of Contents

Section 1 Introduction .....	1
1.1 Mobile text entry .....	1
1.2 Modelling.....	2
1.3 Thesis outline .....	4
Section 2 Less-tap.....	5
2.1 Introduction .....	5
2.1.1 Mobile Phone Keypad.....	5
2.1.2 Mnemonic Numbers .....	6
2.1.3 Eyes-free Input.....	6
2.1.4 Existing Approaches.....	7
2.1.4.1 Multitap.....	7
2.1.4.2 Two-key Input.....	8
2.1.4.3 Dictionary-based Disambiguation .....	9
2.1.4.4 Dictionary-based Disambiguation with Chording.....	10
2.1.4.5 Prefix-based Disambiguation.....	11
2.1.4.6 Miniature QWERTY Keyboards .....	12
2.1.4.7 Fastap .....	13

2.1.5	Keystrokes per Character ( <i>KSPC</i> ) Metric .....	14
2.2	Less-Tap .....	15
2.2.1	Motivation .....	15
2.2.1.1	Non-dictionary Words and International Users.....	15
2.2.2	Letter Frequencies in English .....	17
2.2.3	Less-Tap: a New Technique for Text Entry on Keypads.....	17
2.2.3.1	<i>KSPC</i> and Distribution of Key Strokes.....	18
2.2.3.2	Optimal letter arrangements in other languages .....	19
2.3	Evaluation of Less-Tap.....	21
2.3.1	Apparatus .....	21
2.3.1.1	Hardware .....	21
2.3.1.2	Software .....	23
2.3.1.3	Set of Phrases.....	25
2.3.1.4	Text copy tasks versus text creation tasks .....	25
2.3.2	Participants.....	26
2.3.3	Procedure.....	27
2.3.3.1	Instructions.....	27
2.3.3.2	Treatment of Errors .....	28

2.3.3.3	Eyes-free sessions.....	28
2.3.4	Results .....	29
2.3.4.1	Entry Speed.....	29
2.3.4.2	Error Rate .....	30
2.3.4.3	Errors: Unnecessary Key Presses .....	30
2.3.4.4	Key Repeat Rate .....	32
2.3.4.5	Keystrokes per Character .....	33
2.3.4.6	Subjective Responses of Participants.....	34
2.3.5	Discussion of the results.....	35
2.3.5.1	Entry Speed vs. Sessions.....	35
2.3.5.2	Why is the Improvement Lower than Predicted? .....	36
2.3.5.3	Cognitive Load .....	36
2.3.5.4	Key Repeat Time for Less-Tap and that of LetterWise .....	37
2.3.5.5	Presence of Two Distinct Groups of Users.....	38
2.3.5.6	Performance in the Eyes-free Mode .....	40
2.3.5.7	Comparison with Other Techniques.....	40
2.4	Summary.....	41
Section 3 A Model for non-Expert Text Entry Speed on 12-Button Phone Keypads.....		43

3.1	Introduction .....	43
3.1.1	Existing models for text entry.....	44
3.1.1.1	Keystroke Level Model.....	44
3.1.1.2	Fitts' Law Based Model.....	46
3.1.2	Published experimental results .....	49
3.1.2.1	Study of <i>T9</i> vs. <i>Multitap</i> .....	49
3.1.2.2	Study of <i>Multitap</i> vs. <i>Less-Tap</i> .....	50
3.2	New model for text entry on 12-button keypads.....	51
3.2.1	Fitts' Law .....	51
3.2.2	Time for various key presses .....	52
3.2.3	Key repeat time .....	54
3.2.4	Mental overhead .....	54
3.2.5	Time to enter a character (Movement Model) .....	56
3.2.5.1	Multi-press Input Methods.....	56
3.2.5.2	Predictive Input Methods.....	58
3.2.6	Linguistic Model.....	59
3.2.7	Combining the Models .....	60
3.3	Verifying the Model.....	61

3.3.1	Model Predictions for Various Text Entry Methods .....	61
3.3.2	Comparison with Experimental Data .....	62
3.4	Summary.....	63
Section 4 Conclusions and Future Work.....		64
4.1	Conclusions .....	64
4.2	Future work .....	65
4.2.1	Non-phone uses for the model .....	65
4.2.2	Measuring cognitive load.....	66
4.2.3	Further improvements to the model.....	66
Bibliography.....		68

## List of tables

Table 1: Possible changes to Less-Tap for other languages .....	21
Table 2: KSPC data by session .....	34
Table 3: Speed and improvement for two groups.....	39
Table 4: Comparison between different text entry methods .....	41
Table 5: Existing model predictions for text entry speed (wpm) .....	49
Table 6: Mean entry speeds for novices and experts [James et al. 2001] (wpm).....	50
Table 7: Mean entry speeds observed [Pavlovych et al. 2003] (wpm) .....	51
Table 8: Predictions of the new model for various text entry methods.....	61
Table 9: Model predictions and experimental results for novices.....	62

## List of Figures

Figure 1: The standard 12-key keypad.....	6
Figure 2: MessagEase keypad layout .....	9
Figure 3: Devices that use miniature QWERTY keyboards .....	13
Figure 4: Fastap keypads.....	14
Figure 5: Letter frequencies in English .....	17
Figure 6: Less-Tap Layout.....	18
Figure 7: Frequency of different ‘multistrokes’ .....	19
Figure 8: Letter frequencies in different languages .....	20
Figure 9: Equipment used in the experiment.....	23
Figure 10: Snap shot of the program window.....	24
Figure 11: Entry speed (wpm) by entry method and session.....	29
Figure 12: Error rate vs. entry method and session.....	30
Figure 13: Total number of errors vs. total number of key presses of that type.....	31
Figure 14: Key repeat rate in different sessions .....	33
Figure 15: Time needed for multiple key presses (for non-expert users) .....	53
Figure 16: Coefficient $D_{init}$ as a function of time.....	58

# Section 1

## Introduction

### 1.1 Mobile text entry

Mobile computing is a rapidly developing field. A wide variety of mobile devices are available on the market today – ranging from mobile phones and small hand-held personal digital assistants (PDA's) with rather limited processing capabilities to high-end notebook computers. In addition, a number of wireless network technologies have been introduced in recent years (GPRS<sup>1</sup>, 1xRTT<sup>2</sup>, Bluetooth<sup>3</sup>, 802.11<sup>4</sup>) that enable different kinds of connectivity between the devices. The demand for the input of text or alphanumeric information will become even stronger as accessing the Internet from a mobile device becomes more common.

Most new mobile phones include the capability to send and receive short text messages. Many people use it, and, in fact, during the last decade there has been a phenomenal growth of the number of text messages sent. According to the GSM World Association [gsmworld], in the year 2002, more than 24 billion messages were transmitted each

---

<sup>1</sup> GPRS (General Packet Radio Service) is a data extension of the GSM cellular technology that allows mobile phones to be used for sending and receiving data over an Internet Protocol (IP)-based network.

<sup>2</sup> 1xRTT is a data extension of the CDMA technology – a standard for mobile telephony used in North America and in many parts of Asia.

<sup>3</sup> Bluetooth is a technology to establish a low-cost short-range radio link.

<sup>4</sup> IEEE 802.11x is a set of technologies that allow building wireless local area networks.

month through GSM<sup>5</sup> networks only. 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.

The numbers for text messaging are more modest in North America at the moment. Nevertheless, with different carriers unifying their previously incompatible networks, most of the barriers towards the use of text messaging have recently been removed, and in the near future sending text messages on this continent will probably become as common as in Europe and Asia.

This thesis concentrates on text entry on mobile phones using a 12-button keypad. It does not consider other alternatives such as pen-based input, miniature keyboards, and speech recognition. Pen-based input, which is commonly used in PDA's and in some phones with a PDA function, is very difficult to use with one hand. Miniature keyboards (see section 2.1.4.6) tend to significantly increase the size of the devices. Speech recognition, while often viewed as the ultimate goal for text entry evolution, has not yet reached the level where it can be considered practical.

## **1.2 Modelling**

As the major difficulty in using short text messaging is the text entry on a telephone keypad, various text entry techniques for phones have been and continue to be developed.

---

<sup>5</sup> GSM stands for "Global System for Mobile Communication" (originally, Groupe Spéciale Mobile). This is the most widely used standard for wireless phone communication, which accounts for over 70% of the world market.

Designing new text entry methods for computing systems is usually labour intensive: one typically needs to build a prototype device and to conduct user studies. That leads us to the idea to create a model that could predict the average performance of a newly devised method as accurately as possible without the need to do either of those abovementioned time-consuming tasks.

A major deficiency of previously presented models for text entry is that they concentrate on predicting expert level performance. However, it is commonly known that very few people who use a technique ever become experts, as most people do not use the phone frequently enough to become experts. An attempt has been made to estimate novice performance by assuming that novices visually scan the keyboard before each press. The Hick-Hyman law<sup>6</sup> was used to predict the time for that visual scan [Soukoreff 2002]. However, my own observations of actual novice users indicate that after a short while the users learn the layout and stop or greatly reduce visual scanning behavior. Soukoreff admits that the model does a poor job of modeling novice stylus typing rates on a soft-keyboard [Soukoreff 2002].

---

<sup>6</sup> The Hick-Hyman law uses information theory to quantify the reaction time to stimulus events. The reaction time can be expressed by the equation  $RT = a + bH$  where  $H$  is the information content of the stimulus. Mathematically,  $H$  can be expressed as the logarithm (base 2) of the multiplicative inverse of the probability of the stimulus, or, loosely speaking, the logarithm of the number of choices [Hick 1952].

### 1.3 Thesis outline

Section 2 of this thesis will introduce a new text entry technique for mobile phones, which has better average performance compared to *Multitap*<sup>7</sup>, is simple to implement and allows entry of arbitrary words. The chapter also presents the results of an experimental evaluation of the technique. Section 3 presents a new model for text entry, which can predict text entry speed for novice users. Furthermore, I analyze the new model with respect to some existing models. Finally, Section 4 discusses conclusions and future work.

---

<sup>7</sup> Multitap is the most commonly used technique for text entry in phones. In order to enter a letter with it, a user presses a corresponding key repeatedly until the letter appears (e.g. press '2' once to enter 'a', twice for 'b' and so on). See section 2.1.4.1 for details.

## Section 2

### Less-tap

In this chapter, *Less-Tap*, a novel text entry method for 12-button keypads, is presented. The traditional touch-tone phone keypad is ambiguous for text input because each button encodes 3 or 4 letters. In *Multitap*, the letters are arranged alphabetically within each button. Similar to the conventional *Multitap* method, the new method requires the user to press buttons repeatedly to enter a required letter. In *Less-Tap*, letters are arranged according to their frequency *within* each button. This way, the most common letters require only one key press.

#### 2.1 Introduction

##### 2.1.1 Mobile Phone Keypad

The traditional 12-button keypad consists of ten number keys and two additional symbols: \* and # (Figure 1). The letters are assigned to the keys in alphabetical order. Although there are some variations (see [DialABC] for details), most keypads adhere to the ITU E.161, also known as ANSI T1.703-1995/1999 or ISO/IEC 9995-8:1994, standard.



Figure 1: The standard 12-key keypad.

### 2.1.2 Mnemonic Numbers

The presence of letters on numeric keys is often used to create mnemonic representations of telephone numbers (e.g. 310-BELL for 310-2355 or 1-800-5NUMBER for 1-800-568-6237). Consequently, it is a good idea to maintain the current assignment of letters to keys. However, it is not crucial to maintain the order of letters on each key.

### 2.1.3 Eyes-free Input

A typical text entry task usually has two visual foci of attention (FOA), because the user attends to both the keypad and the display. With practice, the users will memorize the keypad layout and will be able to press buttons without looking at the keypad, thereby removing one FOA. Removing the second FOA is difficult if the text input method includes any kind of disambiguation (e.g., *T9*, *LetterWise*, or *WordWise* – see next section), as it requires the user's attention to monitor the outcome of keystrokes.

Nevertheless, with approaches that are deterministic (e.g. *Multitap*, two-key methods), it is possible to remove that remaining FOA and, thus make the text-entry task completely eyes-free for experienced users.

The only problem introduced by removing all foci of attention will be the inability to monitor entry errors. However, as the number of entry errors tends to decrease with practice, that may not be a big concern for experienced users (it is assumed that the natural redundancy of the language can compensate for some entry errors anyways).

The reduction of visual attention required for text entry is very useful in certain situations such as working in confined environments, experiencing bad lighting conditions, exposure to excessive vibration, wearable computing, etc.

#### 2.1.4 Existing Approaches

In this subsection, a brief description of the existing techniques for text entry for phone keypads will be given.

##### 2.1.4.1 Multitap

Most phones offer *Multitap* as the standard choice for text entry. In order to enter a letter with *Multitap*, a user presses a corresponding key repeatedly until the letter appears (e.g. press '4' three times to enter 'i', '7' four times to enter 's'). A notable difficulty with *Multitap* is the entry of consecutive letters that appear on the same key (*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

with 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 [MacKenzie et al. 2001].

#### 2.1.4.2 Two-key Input

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'). In this approach there is no issue of segmentation. Notable difficulties in this system are how to enter punctuation (the number of commonly used punctuation symbols is significantly greater than three or four) and special characters, as the user has to be able to see *all* the characters mapped to each key to be able to decide which key is to be pressed as the second one.

Another interesting two-key approach, *MessageEase* (Figure 2) is described in Nesbat [2001]. It works as follows:

To enter most frequent 9 letters (E, T, A, O, N, I, R, S, or H): double click a key

(e.g., to enter A, double click key 1)

To enter less frequent 8 letters (U, P, B, J, D, G, C, or Q): (from key 5)

Click 2 keys: key 5 then the key in the direction of the letter

(e.g. to enter U, click key 5, then key 2)

To enter least frequent 8 letters (V, L, X, M, F, W, Y, or K): (to key 5)

Click 2 keys: Its key, then key 5

(e.g. to enter V, click key 1, then key 5)



Figure 2: MessagEase keypad layout

The biggest difficulty with *MessagEase* is that it uses a non-standard layout.

Both two-key methods are not much faster than *Multitap* [Silfverberg et al. 2000].

*Multitap* and the abovementioned two-key entry methods support *eyes-free input* – the ability to enter text without having to visually verify the result – for users who have had an opportunity to memorize the layout of the keypad.

#### 2.1.4.3 Dictionary-based Disambiguation

There are several text entry methods that use a disambiguation 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. Either during the entry, or after the SPACE key is pressed, the system tries to match all possible interpretations of the entered

key sequence to the words that are contained in the dictionary. The most probable word is the default choice offered. If it is not the word intended by the user, he or she has to press a special key (NEXT key) until the intended word appears.

This class of systems fails if the word is not in the dictionary or is misspelled. In such cases, the user will have to either use a sequence of backspaces to correct the error, or re-enter the word from the beginning using another technique, usually *Multitap*. Kober et al. mention that if the commonly used assumptions about *T9*, such as no input errors, are removed, *T9*'s performance degrades significantly [Kober et al. 2001].

There are additional indications that the actual process of entering text using dictionary based methods is somewhat distracting [Gutovitz 2003]. The main complaints are about the unpredictable nature of the system's response to the entry of non-dictionary words. Also, since the user doesn't know if the letters that are shown to him or her on the screen during the entry will ever be 'magically' converted to the intended word, this approach has a relatively high cognitive load, as the user needs to visually verify the result. Finally, eyes-free input is impossible with this class of techniques.

#### 2.1.4.4 Dictionary-based Disambiguation with Chording

*WordWise*<sup>TM</sup>, a method developed by Eatoni Ergonomics, attempts to reduce the drawbacks of the classic word disambiguation by allowing entering one of the letters from each key ('c', 'e', 'h', 'l', 'n', 's', 't', and 'y') unambiguously through the use of an auxiliary key. For example, there are three letters on the button 2. Pressing the button by itself means a user wishes to enter 'a' or 'b', while pressing it while holding the auxiliary

key will always enter 'c'. This reduces the amount of processing that the system has to do and also makes the system more resistant to entry errors [Eatoni]. Some of the words (such as "the" and "then") can be entered unambiguously, which makes it unnecessary to store them in a dictionary.

However, many of the problems of the dictionary-based disambiguation methods, including the issue of non-dictionary words that must be entered with *Multitap* and the impossibility of the eyes-free input, still exist. As well, another problem is introduced – the requirement to enter "chords", as opposed to single keystrokes, which can potentially slow the users down and which makes the one-handed use of the devices much more difficult.

#### 2.1.4.5 Prefix-based Disambiguation

The only known system that uses a prefix-based disambiguation is *LetterWise*<sup>TM</sup>, also by Eatoni Ergonomics. It has been analysed in a longitudinal study [MacKenzie et al. 2001]. The technique works by guessing the most probable next letter based on what the user entered as the beginning of the word – up to three consecutive letters are analyzed in their published implementation. *LetterWise* was designed to avoid the drawbacks associated with the systems mentioned before. Since it uses a database of prefixes (*prefix* is the sequence of letters preceding the current keystroke) 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.

MacKenzie et al. report a 36.6% increase of entry speed for *LetterWise* (relative to *Multitap*) after 10 hours of use [MacKenzie et al. 2001].

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 users intention. Therefore, eyes-free input is not possible.

#### 2.1.4.6 Miniature QWERTY Keyboards

The next two approaches use keypads different from the standard 12-button phone keypads and are described here only to give an overview of other text entry alternatives. They will not be considered further in this thesis.

While the miniature keyboards seem to be a natural choice for text entry, their use in mobile phones is limited, mainly due to their size and the inability to touch type. Also, the single-handed performance for miniature QWERTY keyboards is significantly lower than double-handed performance.



Figure 3: Devices that use miniature QWERTY keyboards (Handspring Treo 300, BlackBerry 6710, Nokia 6800)

#### 2.1.4.7 Fastap

Fastap is described in Levy [2002]. This method 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’, Figure 4 left) or some other similar configuration (Figure 4 right) and the letters are assigned to the buttons sequentially, in alphabetical order. To enter a letter, the user simply presses the corresponding key; to

enter a number or a special symbol, he or she simultaneously presses two buttons located at the opposite corners of some cell of the grid (this is possible if one centres a finger or a thumb between the buttons) – a method the authors call “passive chording”.



Figure 4: Fastap keypads

Each letter requires only one press, just like a QWERTY keyboard and many characters are easily accessible.

I am not aware of the author’s conducting a user study to compare the actual performance of their method against other alternatives.

#### 2.1.5 Keystrokes per Character (*KSPC*) Metric

Keystrokes per character (*KSPC*) is a useful metric for comparisons between two text-entry methods [MacKenzie 2002]. For simplicity, the following discussion will be limited only to lowercase letters and will ignore capitals and special symbols. In this case, *KSPC*

value is equal to one for a standard QWERTY keyboard since there is a dedicated key for each letter. Given a text-entry method and a language corpus, it is straightforward to compute a KSPC value for that method by dividing the number of key presses required by the number of characters in the corpus.

KSPC values for other methods are 1.1500 for *LetterWise*, 1.0072 for *T9*, and 2.0342 for *Multitap* with the use of a timeout kill key [MacKenzie et al. 2001].

It should be noted, however, that the KSPC is not the only parameter affecting the speed of the input. For example, if the layout is unfamiliar and/or confusing, the speed with which the keystrokes are performed will be significantly lower thus affecting the final text entry speed. This will be further investigated later on.

## **2.2 Less-Tap**

### 2.2.1 Motivation

The motivation behind the design of *Less-Tap* was a text entry technique for mobile phones that would be easy to use, easy to learn, easy to implement, yet still be compatible with a standard keypad layout. Obviously, to be of any use, the technique would have to be faster than *Multitap*.

#### 2.2.1.1 Non-dictionary Words and International Users

One interesting fact is that the language commonly used in short text messaging is often quite different from the language that is used to write books [MacKenzie et al. 2001].

This poses a major problem for dictionary-based disambiguation methods, as they limit the user to contents of a dictionary, more precisely to the dictionary built into the device. Words outside the dictionary can be handled only by switching dictionaries or by using *Multitap*, for example. Another issue is the fact that ethnic minorities in many countries often want to communicate between themselves in their language, which may not be supported by the devices that they can buy. Similarly, there are some nations in the world whose population may be too small to warrant localization of some particular communication device. One of the examples of small nations is Iceland, with a population of just several hundred thousand.

In these last two cases, *Multitap* is currently the only option for text entry. The technique described here, *Less-Tap*, was designed to work best with English. However, it should work well in the two situations described above, possibly with a small drop in performance (see section 2.2.3.2 for details). For fairness, it must be acknowledged that *LetterWise* also deals well with words from languages that were not considered when building its database.

### 2.2.2 Letter Frequencies in English

The arrangement of letters on the keys in *Less-Tap* depends entirely on letter frequencies in English. In this paper, the data was derived from the British National Corpus<sup>8</sup> [BNC].

Figure 5 below shows the computed frequencies.

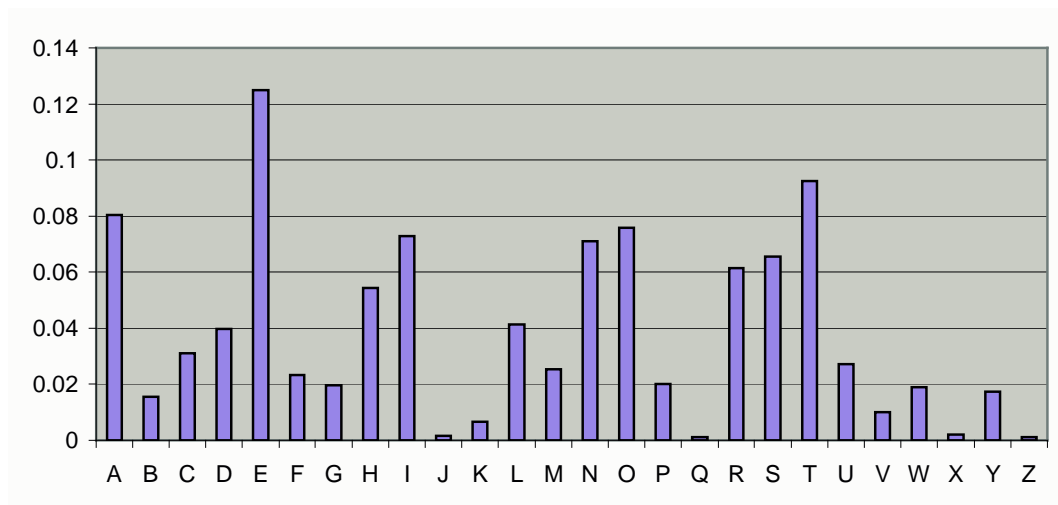


Figure 5: Letter frequencies in English

### 2.2.3 Less-Tap: a New Technique for Text Entry on Keypads

*Less-tap* differs from *Multitap* in the order in which the letters appear upon pressing a key. The objective was to allow the entry of the most frequent letter on each key with one

---

<sup>8</sup> It is important to note that the British National Corpus is not necessarily the best choice for a corpus here. Ideally, a corpus of short text messages (SMS) should be used. However, there is currently (May 2003) no publicly available corpus of SMS messages and it is unclear how well SMS corresponds to other sources such as ICQ (internet chat). Consequently, I decided to use the British National Corpus, as it allows for comparisons with other studies.

keystroke, the second most frequent letter with two keystrokes and so on. After using the data on letter frequencies (Figure 5) the following solution was obtained (see Figure 6).

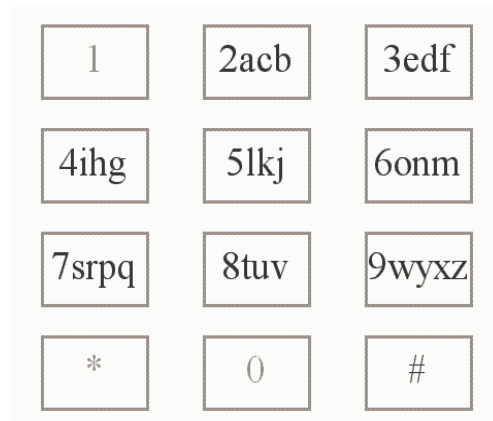


Figure 6: Less-Tap Layout.

By keeping the letters on the same keys, it is ensured that the keypad remains compliant with keypad standards and that the mnemonic numbers can still be entered.

#### 2.2.3.1 KSPC and Distribution of Key Strokes

As mentioned earlier, the timeout kill key was not used in the user tests for this work. Without the timeout kill key, the computed value for KSPC is 1.9488 for *Multitap* and 1.4412 for *Less-Tap*. For *Less-Tap* with a timeout kill key the KSPC value is 1.5266 and 1.9488 for *Multitap*.

To further illustrate the effect of our re-arrangement of the letters, I also computed the distribution of the number of key presses required to enter the complete text of the British National Corpus.

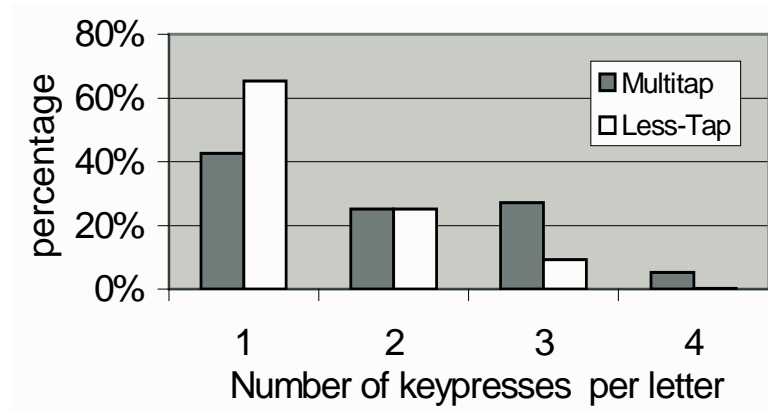


Figure 7: Frequency of different 'multistrokes'.

As one can see from Figure 7, *Less-Tap* lets the user enter about two-thirds of all characters (65.4%) with a single key press. The number of double presses is similar to that of *Multitap*, but there are fewer triple and extremely few (0.11%) quadruple key presses.

#### 2.2.3.2 Optimal letter arrangements in other languages

A short assessment of letter frequencies in some other Latin alphabet-based languages was conducted [Pommerening 2003, TrinColl 2002, UND 2000]. Figure 8 illustrates the relative frequencies of letters in some commonly used languages. Note however, that in general the amount of text sampled was much smaller than the British National Corpus.

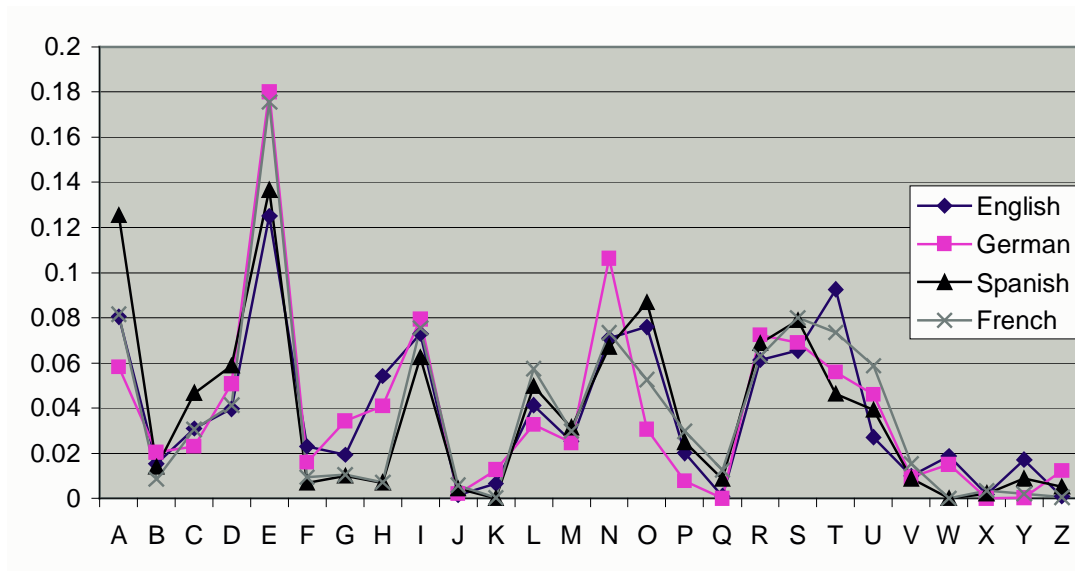


Figure 8: Letter frequencies in different languages

Even though *Less-Tap* was designed with English in mind, it is almost optimal for other common languages as well. Table 1 below summarizes the changes that one should make to make the layout optimal for a given language. In general, the ordering changes on one or two buttons at most. In some cases, the impact of those changes will be barely noticeable (e.g. G and H have comparable low probabilities in both Spanish and Italian).

<i>Language</i>	<i>Changes</i>
Spanish	IHG → IGH
French	ONM → NOM
German	ONM → NOM, SRPQ → RSPQ
Italian	IHG → IGH, SRPQ → RSPQ

Table 1: Possible changes to Less-Tap for other languages

## 2.3 Evaluation of Less-Tap

So far, it has only been established that *Less-Tap* requires significantly fewer keystrokes per character in the ideal case – when no errors are made and the words entered are those from the British National Corpus. In this section an empirical evaluation of *Less-Tap* is presented which demonstrates how the technique performs in the real world. As in previous studies [MacKenzie et al. 2001], *Multitap* was used as a comparison point.

### 2.3.1 Apparatus

#### 2.3.1.1 Hardware

In this study, I used an actual Nokia 5190 handset. To implement the techniques, I connected the keypad of the mobile phone to the keyboard of a PC. Here I exploited the fact that both keyboards are based on a matrix layout, i.e. horizontal and vertical rows. I electrically connected the 4x4 grid of the Nokia keypad to a similar part of the PC keyboard (the region delimited by the keys ‘1’, ‘4’, ‘v’, and ‘z’) with a lightweight 8-wire

cable. In effect, each key press on the Nokia keypad entered a character into the PC. With this, I was able to maintain the form factor, weight, size, as well as the typical 'feel' of a mobile phone. The buttons were *not* re-labelled to reflect the different layouts of the letters. While re-labelling could increase the performance for the new method, it was decided to keep the *hardware* modifications of the phone to a minimum. The 5100-series models are relatively common which should make it easy to replicate this study. In the experiment, only the 12 standard keys as shown in Figure 1 were used.

Since displaying the characters on the LCD display of the (non-working) handset is hard to achieve for technical reasons, I chose to display the entered text on a Wacom LCD tablet. The advantage of using a tablet is that it allowed users to find quickly the most convenient position for their text entry sessions. For example, participants could hold the handset right in front of the slanted display beside the output window where the text that they were entering was displayed. Some participants chose to hold the handset on their laps or in the air.



Figure 9: Equipment used in the experiment.

#### 2.3.1.2 Software

The software that was used in the experiment was written in Java and had been used previously for text entry experiments [MacKenzie et al. 2001]. It was modified to interpret the characters entered on the phone keypad connected to the PC keyboard. Furthermore, the option to use either *Multitap* or *Less-tap* (referred to as Alternative Text Entry Method during the experiment) was added. Figure 10 shows the user interface.

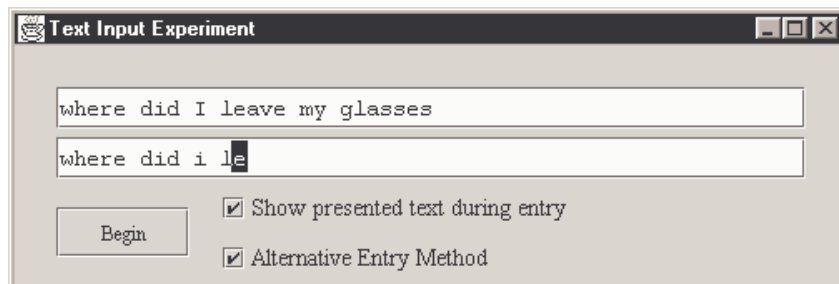


Figure 10: Snap shot of the program window.

The first text field shows the phrase to be entered and the second field shows the text that the user has entered so far. The software recorded all key presses and the times when they took place. Furthermore, the software did statistical calculations such as the time to enter a phrase, error rate, average WPM speed for the phrase, key repeat time and so on and displayed these values at the bottom of the window for control purposes.

The buttons that don't have letters on them were assigned the following actions: '\*' – backspace, '1' – end of phrase, '0' and '#' – both space.

The timer started with the first typed key for each phrase and stopped with the '1' key so that the participants could rest between the phrases at their discretion. They were informed about this feature of the system.

As I was testing first time users, a timeout kill key was not provided in the design. This decision is based on indications that the use of a timeout kill key has a noticeable advantage only for expert users [Silfverberg et al. 2000]. The timeout was set to 1.0 s. The text cursor in the software changes from block ('overwrite') to line ('insert')

depending on whether the timeout had expired or not. In *Multitap* and *Less-Tap*, pressing the same button within the timeout changes the last letter entered.

I also used a second version of the program in order to test my technique in “eyes-free” mode. In that version, the second text field as in Figure 10 was completely blanked out (even the cursor was not visible).

#### 2.3.1.3 Set of Phrases

The set of phrases was the same as the one used in the test of *LetterWise* and is representative of English [MacKenzie et al. 2001]. It was later published in [MacKenzie et al. 2003]. The phrases to be entered were chosen randomly from this file.

#### 2.3.1.4 Text copy tasks versus text creation tasks

An important decision in text entry evaluations is whether to use text copy tasks or text creation tasks. In a text copy task, the participant is *given* text to enter using the input technique under investigation. In a creation task, the source text is either memorized or *generated* by the subject.

The main advantage of a text creation task is that it imitates typical usage. However, it has many disadvantages [MacKenzie et al. 2002]:

- The first one is identifying errors – it is difficult to know exactly what a subject intended to enter if the subject is generating the text.
- A second is the loss of control over the distribution of letters and words entered.

The task should require the subject to enter a representative number of

occurrences of characters or words in the language (in order for the results to be generalizable). Yet, it is not possible to control this if the subject is generating the text.

- A third is the measurement of text entry speed. Measurements will include the time invested in pondering – thinking about what to write. It is difficult to separate this from the effort of actually entering the text.

The disadvantages just cited are significant and are the reason that most researchers prefer text copy tasks and the rationale for me to do the same.

### 2.3.2 Participants

There were 12 participants in the test who were recruited through advertisements posted on the university campus. A pilot study indicated that this number might be needed to achieve the statistical significance and other studies in the literature have used similar numbers. Three participants were female, one was left-handed, and three were frequent users of text messaging. The ages ranged from 19 to 39 with the mean of 24.3. All but two had extensive computer experience (five years and more). One did not own a cell phone. Three had reported using text messaging on the cell phone weekly or daily; another three used it monthly. All participants were paid \$20 upon completion of the user study.

### 2.3.3 Procedure

Each participant was assigned three time slots, 30–40 minutes each, with short breaks between them. Each slot contained one session with each method, 20 phrases each. A within-subjects design was used and the order of entry methods was counterbalanced to compensate for learning effects<sup>9</sup>.

#### 2.3.3.1 Instructions

Before the test, the participants were given brief instruction as to their task, how the software worked, as well as which keypad buttons were doing what. They were told that they were free to hold the handset in any way they chose during the test. They were also given the freedom to adjust the position and the orientation of the LCD screen.

The participants were briefly instructed on how to enter text with *Multitap*. For *Less-tap*, it was also explained that the sequence in which the letters appeared on the screen for that system was different from that of *Multitap*. Before each session, users were encouraged to enter a few phrases for practice and to help to transition from the previous technique to the next.

Two small paper sheets illustrating the letter layout for both methods were available (similar to Figure 6). These sheets were normally placed at the bottom-left corner of the screen.

### 2.3.3.2 Treatment of Errors

As to errors, the participants were told to pretend that they were typing a message to their friend so that one or two typos per phrase were OK. In other words, I did not force the participants to enter the phrases completely error-free. They were told that the errors could be corrected via a backspace (‘\*’) button. Also, if they pressed a correct key but a wrong number of times, they could continue pressing it until the intended letter appeared for the second (or third) time.

There were occasional “uncorrectable” errors when users pressed the “end of phrase” button in the middle of the phrase. In those cases, the participants were told “not to panic” and were simply asked to enter some additional phrases at the end of the session.

### 2.3.3.3 Eyes-free sessions

In their third session, the participants were asked to enter the phrases without visual feedback (they could see the text that they had to enter but they were not able to observe the progress). They were also asked to try not to look at the button assignment sheet and the buttons themselves.

---

<sup>9</sup> *Within-subjects design*, as opposed to *between-subjects design*, means that each participant tested both methods. *Counterbalancing* is used in within-subjects designs to control learning effects. E.g., in this study, half of the participants tested the techniques in sequence AB-AB-AB while the other half did BA-BA-BA.

## 2.3.4 Results

### 2.3.4.1 Entry Speed

Overall, the mean entry rate was 7.15 wpm ( $\sigma = 0.18$ ) for *Multitap* and 7.82 wpm ( $\sigma = 0.18$ ) for *Less-Tap*. In other words, *Less-tap* was faster by 9.5%. The main effect for entry method was statistically significant<sup>10</sup> ( $F_{1,11} = 7.32, p < 0.05$ ). Figure 11 demonstrates the entry speed in words per minute for different sessions.

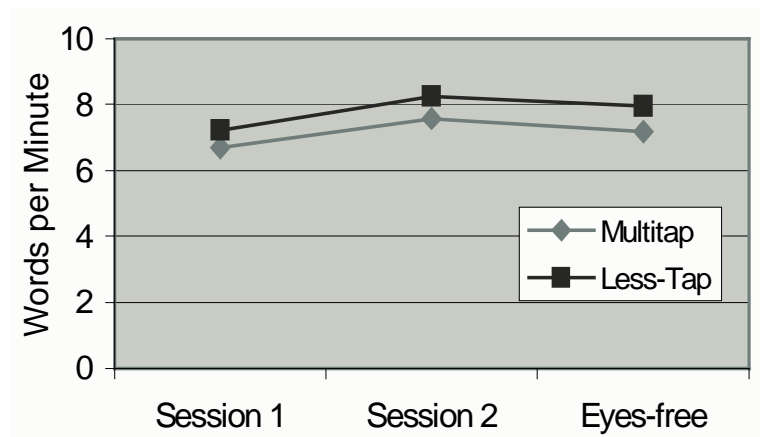


Figure 11: Entry speed (wpm) by entry method and session.

---

<sup>10</sup> The F-number is defined as the ratio of the variances between and within groups:  $F = \sigma_{\text{between}}^2 / \sigma_{\text{within}}^2$ . The indices indicate the number of degrees of freedom between and within groups respectively. The p-number can be interpreted as the probability of the distributions being statistically similar. The difference between the distributions is considered to be statistically significant if  $p < 0.05$ .

#### 2.3.4.2 Error Rate

Over all sessions, the difference in error rate was not statistically significant between *Multitap* and *Less-Tap* (Error Rate = 0.73 resp. 0.74,  $\sigma = 0.095$ ,  $F_{1,11} = 4.66$ ,  $p > 0.05$ ). However, the difference was significant for the eyes-free session (Error Rate = 6.66 resp. 3.97,  $\sigma = 0.86$ ,  $F_{1,11} = 4.91$ ,  $p < 0.05$ ). Figure 12 shows the graphs for error rate for each session.

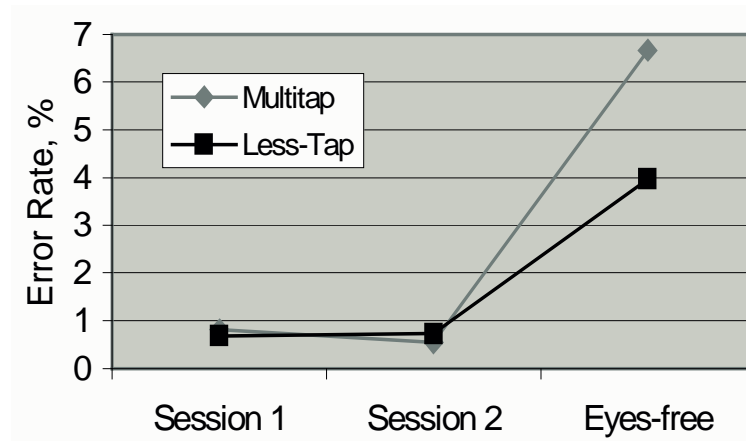


Figure 12: Error rate vs. entry method and session.

#### 2.3.4.3 Errors: Unnecessary Key Presses

To analyze the errors further, I computed the number of unnecessary key presses. This type of error occurs when the user presses a key too often. Then they have to press them a few more times to get the intended character. While it was possible to correct those errors with the backspace button, almost all users preferred to continue pressing the same button

thus going for another round (e.g. the letter ‘a’ can be entered by pressing the button ‘2’ one, four, seven or more times).

The data for the following figure were obtained by analyzing the key press log data. The absolute number of unnecessary key presses was about the same for both methods. What differed was the distribution of the errors among different types of key presses (single, double etc.). To illustrate this, I computed the percentage of extra key presses in the total number of key presses of each type. The resulting graph is shown in Figure 13. For example, if 2700 key presses were required to enter all letters that need two key presses each, and 3000 were actually performed, then the percentage of extra presses would be 10%.

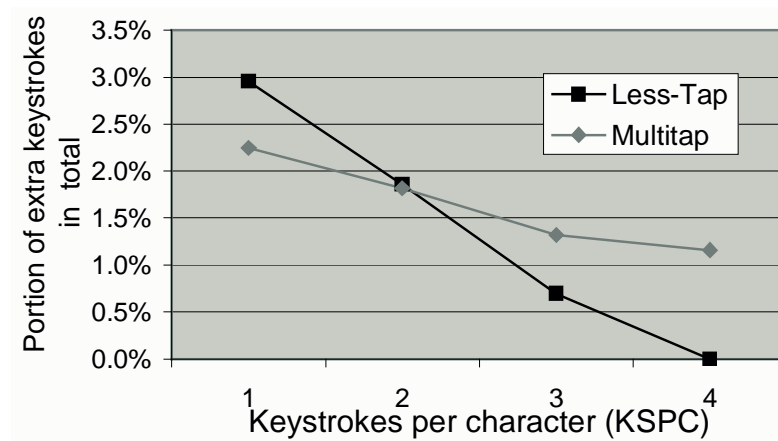


Figure 13: Number of extra key presses vs. total number of key presses of that type.

The graph shows that the number of errors is significantly greater for single key presses in *Less-Tap*. I have no real good explanation for this, but it is likely to be related to the

distribution of the types of key presses in both systems (Figure 7 on page 19). It could be possible that users expect to make at least two button presses for each letter (which is reasonable, considering that most buttons contain three letters and that the average number of keystrokes per character in *Multitap* is close to two). Yet, as most of the letters in the new system require only one press, such strategy would cause frequent “overshoots”. Similarly, as letters that require triple and quadruple keystrokes are very rare in *Less-Tap* (less than 10%), users would tend to pay more attention to entering these letters in *Less-Tap*, thereby reducing the number of mistakes that they make when entering such letters.

#### 2.3.4.4 Key Repeat Rate

The key repeat rate, i.e. how fast keys were hit, was lower for *Less-Tap* (1.04 vs. 1.27 keystrokes per second). The main effect for entry method was statistically significant (KSPS = 1.27 resp. 1.04,  $\sigma = 0.022$ ,  $F_{1,11} = 53.97$ ,  $p < 0.05$ ). Figure 14 shows the graphs by session.

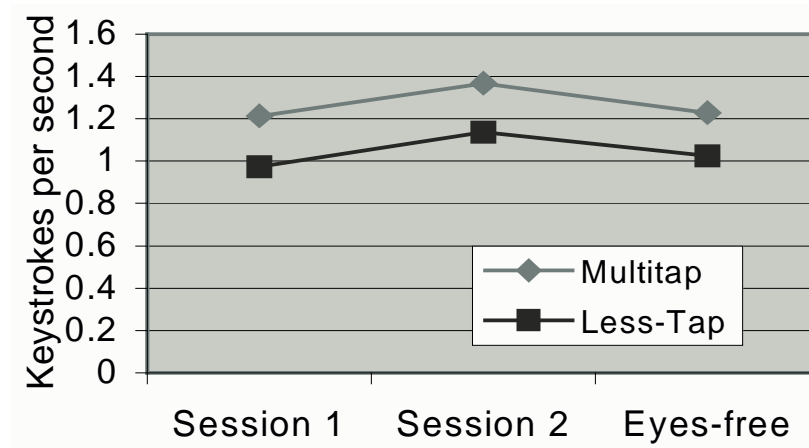


Figure 14: Key repeat rate in different sessions

As far as I can determine it, the fact that *Less-Tap* always had fewer keystrokes per second (KSPS) than *Multitap* can be attributed to different amount of cognitive load involved in the two techniques. Section 3 is dedicated to the analysis of this issue.

#### 2.3.4.5 Keystrokes per Character

The difference in the observed number of keystrokes per character was highly significant between methods (KSPC = 2.15 resp. 1.62,  $\sigma = 0.020$ ,  $F_{1,11} = 332.74$ ,  $p < 0.001$ ).

Session	Multitap	Less-Tap
1	2.1904	1.6254
2	2.1927	1.6808
3 (eyes-free)	2.0728	1.5660
All	2.1505	1.6215
Predicted (see section 2.1.5)	1.9488	1.4412

Table 2: KSPC data by session

For both methods, the values were 10–15% greater than those predicted. This increase is caused partially by the unnecessary key presses, as discussed in section 2.3.4.3, as well as by use of wrong keys or the backspace key. Overall, *Multitap* required about 33% more keystrokes, which (almost) matches the prediction (35%).

#### 2.3.4.6 Subjective Responses of Participants

After the test, the users were asked to fill out a short questionnaire that contained several questions that were rated on a seven-point Likert scale<sup>11</sup> (from -3 to +3). For example, the choices for perceived speed were ‘much faster’, ‘faster’, ‘slightly faster’, ‘no difference’, ‘slightly slower’, ‘slower’, and ‘much slower’.

---

<sup>11</sup> In 1932, Renis Likert invented a measurement method, called the Likert Scales, designed for use in attitude surveys [Likert 1932]. Such scales allow answers that range from “strongly disagree” to “strongly agree” and are considered the standard for questionnaires in human-computer interaction research.

Summarizing the responses, in comparison to *Multitap* users felt that the new system was:

- faster (average score +1.6)
- slightly easier to use (+1.3)
- slightly less tiring (+1.2)
- slightly easier to learn (+0.9)
- somewhat preferred, in that they wanted *Less-Tap* as a permanent replacement of *Multitap* in their phone (+1.3).

Most participants commented that they would prefer *Less-Tap* more if the buttons were re-labelled to reflect the changed ordering. The results of the subjective responses are not statistically significant.

## 2.3.5 Discussion of the results

### 2.3.5.1 Entry Speed vs. Sessions

It can be seen that speed improves as time using the new system increases; both from the improvement of key repeat time (Figure 14) as well as the improvement of text entry speed in WPM (Figure 11).

One important factor here is that different people get tired at different times. While some can perform a repetitive task for hours, others get bored after 20 minutes. In the experiment, I asked participants to work in one-hour slots. The analysis of the log files

indicated that by the end of their respective time slot, speed dropped for some of the users.

#### 2.3.5.2 Why is the Improvement Lower than Predicted?

Since the predicted difference in KSPC values between *Multitap* and *Less-Tap* is 35.2%, the question arises as to why the improvement to the text entry speed is only 9.5%?

The speed of text entry as measured in words per minute can be decomposed into the speed with which the key presses are made, known as a key repeat rate (in keystrokes per second), and the average number of keystrokes required to enter characters. In this thesis, 'word' is defined to have five characters on average; a convention commonly used in the literature on text entry methods.

The results above showed that the discrepancy between the actual and predicted values for KSPC is similar for both techniques. As I can observe from Figure 14, the key repeat rate is significantly lower for *Multitap* than it is for *Less-Tap*. It took on average about 20% longer to perform a button press in *Less-Tap*. I hypothesize that the difference is due to the way the cognitive load is distributed between different key presses in the two techniques (for example, in the case of a triple key press, the first press will be preceded by a significantly longer delay than the other two). This will be analyzed in Section 3.

#### 2.3.5.3 Cognitive Load

Cognitive load is usually described as the additional effort or concentration in keeping track of several things (such as positions of letters on keys) at a time [Conklin 1987]. In

the experiments above, the cognitive load is interpreted as the additional time that precedes the keystrokes.

While it was expected that *Less-Tap* would be about as easy to learn as *Multitap*, there may have been some differences which caused the users to be slower in learning or using *Less-Tap* (even though they thought the opposite and actually showed higher entry rates). Initially, before the work described in Section 3 was done, these differences were attributed solely to additional mental processing required by *Less-Tap*, as it was thought that *Less-Tap* was more complex and harder to learn due to the non-alphabetical arrangement of letters. Yet, considering that, overall, participants found *Less-Tap* “easier to use” and to be “slightly less tiring”, it may be concluded that the increased cognitive load in *Less-Tap* caused less fatigue than more button presses in *Multitap*.

Finally, it should be noted that the speed at which key presses were made was significantly lower than theoretically predicted (see [Silfverberg et al. 2000], for example) – in both systems. Thus, it is likely that, with further training, the advantage of *Less-Tap* over *Multitap* will increase, as the amount of cognitive load tends to diminish with practice.

#### 2.3.5.4 Key Repeat Time for Less-Tap and that of LetterWise

It would be interesting to compare the findings mentioned above to those of alternative systems, notably, *LetterWise* [MacKenzie et al. 2001]. While there are no data that state a key repeat rate for *LetterWise*, there is a very detailed graph [MacKenzie et al. 2001, Figure 6) showing the entry speed in WPM by session for both *Multitap* and

*LetterWise*. The authors of that paper did a longitudinal study (10 hours with each system) and they used a desktop size keypad. I estimated from their graph the values for the third session (approx. 1.5 hrs into the test vs. 1 hr in our case). The values obtained were 9.1 and 10.5 WPM for *Multitap* and *LetterWise* respectively. That is, *LetterWise* is about 15% faster than *Multitap* after 1.5 hours of use. However, the difference is much larger in terms of KSPC: 2.03 vs. 1.15 respectively (about 77%). This means that, with *LetterWise*, it takes about 54% ( $1.77 / 1.15$ ) longer to perform key presses than with *Multitap*. Since *Less-Tap* required only about 20% more time for each key press than *Multitap* (see Figure 14), I may conclude that the cognitive load of *Less-Tap* per key press is smaller than that of *LetterWise*. No similar data could be found on T9.

#### 2.3.5.5 Presence of Two Distinct Groups of Users

When analyzing the results, it was discovered that there seemed to be two distinct groups of participants, each accounting for exactly half the participants. In one group, *Less-Tap* is much faster than *Multitap*, almost 20% ( $\sigma = 5.2$ ), with high statistical significance (7.12 resp. 8.54 wpm,  $\sigma = 0.11$ ,  $F_{1,5} = 83.87$ ,  $p < 0.001$ ). In the other group, the two methods were essentially the same in terms of entry speed (0%,  $\sigma = 6.3$ ) and error rate (for entry speed, 7.17 resp. 7.11 wpm,  $\sigma = 0.12$ ,  $F_{1,5} = 0.13$ ,  $p \gg 0.05$ ). In the other group, three users performed slower with *Less-Tap* but for only one of them was the difference greater than 2%, and in that case it was caused by poor performance with *Less-Tap* during the “eyes-free” session. The difference between the groups in terms of the

improvement of the entry speed is statistically significant ( $F_{1,5} = 37.25, p < 0.001$ ). Table 3 below shows the entry speed for each user.

From this, it can be speculated that some people can easily adapt to re-ordered letter sequences, while others find them significantly harder to memorize and take much longer to learn them. However, I believe that with more use eventually most people could learn the new system and benefit from it. A long-term study would be needed to analyze this further.

User number	Speed in Multitap, wpm	Speed in Less-Tap, wpm	Improvement
<i>Group 1</i>			
6	7.65	6.88	-10%
9	6.28	6.15	-2%
1	9.90	9.71	-2%
8	7.01	6.98	0%
11	7.11	7.44	5%
4	5.06	5.48	8%
<i>Group 2</i>			
7	8.77	10.08	15%
5	7.77	9.00	16%
3	6.06	7.10	17%
2	5.59	6.81	22%
12	7.35	8.97	22%
10	7.21	9.28	29%

Table 3: Speed and improvement for two groups

#### 2.3.5.6 Performance in the Eyes-free Mode

It can be seen from the figures above that, in the eyes-free mode, *Less-Tap* performs no worse than *Multitap*. And, in fact, users make fewer errors with *Less-Tap* in eyes-free mode (Figure 12 on page 30). Of course, more practice is required in order for the error rate (approximately 4%) to drop to a level that is comparable to the error rates for text entry with visual feedback (less than 1%). Exploring the natural redundancy of the language would also help here.

#### 2.3.5.7 Comparison with Other Techniques

There are several methods to enter text on the mobile phone keypad. The following table briefly summarizes the advantages and disadvantages of different techniques for 12-button keypads. As one can see, none of the techniques is ideal. A technique that is fast (such as *T9*) is very restrictive at the same time. Techniques that are not restrictive (*Multitap* and *Less-Tap*) require significantly more than one key press per character. Based on the table data, *LetterWise* looks like a good compromise between speed and unobtrusiveness. However, it does not allow eyes-free input. Note that the only weakness of *Less-Tap* with respect to *Multitap* is the non-alphabetical ordering of the letters on keys. Yet, it has been demonstrated that this was not an obstacle for users.

Method	Multitap	Less-Tap	LetterWise	T9
Property				
Available on mobile phones	Almost all	N/A	None known	Many newer models
Enforces spelling	No	No	No	Yes
Non-dictionary words	Yes	Yes	Yes	No
Eyes-free input	Yes	Yes	No	No
Implementation *	Very easy	Very easy	Moderate	Hard
KSPC	1.9488 or 2.0342 (see text)	1.4412 or 1.5266 (see text)	1.1500	1.0072* <sup>2</sup>
Adaptation to other languages	Do nothing	Nothing or change letter ordering	Nothing or change database	<i>Must</i> change dictionary

\* – includes memory and processing requirements

\*<sup>2</sup> – may be significantly greater depending on the corpus

Table 4: Comparison between different text entry methods

## 2.4 Summary

A new technique to enter text on mobile phone keypads, which requires 25% fewer keystrokes compared to *Multitap*, was presented.

Unlike dictionary based methods, *Less-Tap* facilitates the entry of arbitrary words. Unlike *LetterWise* and *T9*<sup>®</sup>, *Less-Tap* allows entering text without having to visually verify the

result (eyes-free input), after some initial training. For English, *Less-Tap* requires an average of 1.4412 keystrokes per character (vs. 1.9488 in *Multitap*).

A user study to compare *Less-Tap* against *Multitap* was conducted. Each participant had three 20-minute sessions with each technique. The results of the user study show that for first-time users the new technique is on average more than 9.5% faster.

*Less-Tap* is very easy to implement, like *Multitap*.

## **Section 3**

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

### **3.1 Introduction**

Various text entry techniques for phones have been developed. Designing new text entry methods for computing systems is usually labour intensive: one typically needs to build a prototype device and to conduct user studies. Thus, the existence of a model that would predict the performance of a new method as closely as possible without the need to do either of these time-consuming tasks is considered to be very valuable.

In this chapter, a new model for predicting text entry speed on a 12-button mobile phone keypad is presented. The model is designed to 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. The contribution of the model is that it adds a mental preparation time before key presses and that it considers the fact that multiple presses of the same key cannot be modelled accurately by Fitts' law. Finally, the prediction of the model is compared to previously published experimental results.

### 3.1.1 Existing models for text entry

At the time of writing, two models for text entry on a 12-button keypad were known. Both of them were designed to predict expert (or *peak*) text entry rates for various text entry methods.

#### 3.1.1.1 Keystroke Level Model

The first model – a keystroke level model by Card et al. [Card et al. 1980; Card et al. 1983, Chapter 8] is one of the earliest predictive models for human-computer interfaces. This model is general and can be applied not only to text entry tasks but also to mouse pointing, drawing etc. The model predicts the task execution time by trying to account for times of different actions or events, such as key press, pointing to a target with a mouse, homing hands on a keyboard or a mouse, drawing a line with a mouse, mental preparation time (pause), system response time, etc.

A researcher who intends to use a model would use only the components that are relevant in the circumstances. Dunlop and Crossan have used this model to compare the theoretical speed of their text entry technique (a predictive method similar to *T9*) to that of *Multitap* [Dunlop et al. 2000]. They used the following factors in their modeling (taken from [Card et al. 1980] and based on human performance data):

$T_k$  – The time to physically press a button. The value used was 0.28 s, which corresponds to “an average non-secretary typist” in the original model.

$T_h$  – Homing time for the hand to move to the keyboard. This was fixed at 0.4 s.

$T_m$  – Mental preparation time for executing a physical action (1.35 s).

For *Multitap*, they assume that the time to enter a given phrase ( $P$ ) involves a homing action followed by the entry of  $w$  words. Each word contains  $k_t$  key presses on average and  $d$  delays. Average word length is assumed to be 5.98 characters, which was derived from their experimental text (newspaper articles), the average number of key presses per word is 11.85 and the average number of *Multitap* timeouts was computed at 0.49 per word. Therefore, the model for predicting *Multitap* speed is:

$$T(P) = T_h + w(k_t T_k + dT_m)$$

Predictive text input is assumed to have only one key press per character, so that the number of keystrokes per word,  $k_p$  is equal to the average word length. Each word also requires  $l$  presses of the end-of-word key, on average, to select the correct prediction (in the paper,  $l = 1.03$ ). Then, the model for predictive input is<sup>12</sup>:

$$T(P) = T_h + w(k_p T_k + l(T_m + T_k))$$

In their paper, the model gave predictions of 14.9 and 17.6 words per minute (wpm) for *Multitap* and the predictive text entry method respectively. I recomputed the predictions

---

<sup>12</sup> I believe that the equation in [Dunlop and Crossan, 2000] 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. I corrected it by replacing the factor  $k_p$  by  $(word\ length - 1)$ .

of the model considering that other models assume an average word length of 5 and the issue mentioned in the footnote. The revised predictions appear in Table 5.

Major deficiencies of the 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. That 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  $T_m$  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.

#### 3.1.1.2 Fitts' Law Based Model

The second model was presented by Silfverberg et al. [Silfverberg et al. 2000] and is largely based on the model of Soukoreff et al. for soft keyboards [Soukoreff et al. 1995]. It contains two parts: a movement model based on Fitts' law [Fitts 1954] 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.

Fitts' law (see section 3.2.1) is used to determine the time  $MT$  to move from the previous key to the current one.

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

Constants  $a$  and  $b$  are determined empirically for a given device and the interaction method, and  $ID$  is based on the relative positioning and sizes of the key pairs involved (in this model,  $a = 176$  and  $b = 64$  ms/bit for a thumb on a Nokia 5100 series).

A character generally can involve more than one key press. Therefore, a time to enter a character  $CT$  is computed as follows:

$$CT = \sum MT_k$$

where  $MT_k$  are the times for each movement required.

For *Multitap* with timeout, the character entry time is:

$$CT = MT_0 + N \cdot MT_{\text{repeat}} + [T_{\text{timeout}}]$$

$MT_0$  is the initial movement time, i.e. a time to move one's finger from the first key of the digraph to the second.

$N$  is the number of key repetitions (e.g. 1 to enter 'b' with button 2).

$MT_{\text{repeat}}$  is the key-repetition time, an intercept  $a$  of the Fitts' law equation (as the index of difficulty is zero).

$T_{\text{timeout}}$  is included only if the current character is on the same key as the previous one (the value used in the paper is 1.5 s).

With *T9*, the assumption is made that a user would never need to cycle through the predicted words (perfect disambiguation). Consequently, the model states that each letter requires only one key press and makes the equation for *T9* very simple:

$$CT = MT_0$$

A second component of the model is the 27-by-27 table of letter-pair frequencies  $P_{ij}$  in common English (space is the 27<sup>th</sup> character). A different language would require a different matrix of frequencies.

Now, the two components can be combined to obtain the average character entry time for a language L, in seconds:

$$CT_L = \sum \sum (P_{ij} \cdot CT_{ij})$$

where  $i$  and  $j$  range from 1 to 27 in our case.

This value can then be converted to the entry speed in words per minute as follows (assuming an average word size of 5 characters):

$$WPM = (60/5) \cdot (1/CT_L)$$

The 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 thesis, I have recomputed

the numbers to reflect the fact that in my experiments I was using a timeout of 1 s instead of 1.5 s. The revised predictions appear in Table 5.

<i>Method</i>	<i>Model by Card et al.</i>	<i>Model by Silfverberg et al.</i>
<i>Multitap</i>	18.35	22.3
<i>Less-Tap</i>	23.47	26.8
<i>T9</i>	24.97	40.6

Table 5: Existing model predictions for text entry speed (wpm); revised numbers, see text

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

### 3.1.2 Published experimental results

#### 3.1.2.1 Study of *T9* vs. *Multitap*

An interesting observation comes from [James et al. 2001], where the authors evaluate *T9* and *Multitap* in an experiment with both novice and expert users. The results that they obtained are presented in Table 6.

<i>Method</i>	<i>Novice, all</i>	<i>Expert, all</i>	<i>Novice, chat only</i>	<i>Expert, chat only</i>	<i>Novice, newspaper only</i>	<i>Expert, newspaper only</i>
<i>Multitap</i>	7.98	7.93	10.37	10.53	5.59	5.33
<i>T9</i>	9.09	20.36	10.98	25.68	7.21	15.05

Table 6: Mean entry speeds observed for novices and experts [James et al. 2001] (wpm)

As one can see, the actual observed rates do not agree with either model (see Table 5). Also, in the same paper, the authors assert that the notion of an *expert* text message user is unrealistic due to the very nature of the task. The paper reports that a typical SMS user sends only about 20 messages per month – which seems to be consistent with what I were able to observe. As the messages themselves are short, there would hardly be an opportunity for the users to ever become experts!

### 3.1.2.2 Study of *Multitap* vs. *Less-Tap*

In my user study (see Section 2), I was able to observe an improvement of *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 prediction of the first model was 28% and the prediction of the second model was 20%), the actual text entry speeds that I was able to observe were much lower than predicted. Please refer to Table 5 for comparison.

<i>Method</i>	<i>Entry speed</i>
<i>Multitap</i>	7.15
<i>Less-Tap</i>	7.82

Table 7: Mean entry speeds observed [Pavlovych et al. 2003] (wpm)

Trying to explain and to predict the observed discrepancies between previous models and the experiments has led me to the development of a new model.

### 3.2 New model for text entry on 12-button keypads

The model that will be described here can largely be viewed as an extension of the aforementioned model of Silfverberg et al. However, the new model is also applicable to non-expert users. This is a useful extension, since, as stated above, few people ever reach the expert level.

#### 3.2.1 Fitts' Law

Fitts' model is an important component of my model. Fitts' law [Fitts 1954] is a model for 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. distance between buttons on a keypad), and  $W$  is the width of a target (e.g. a button, in our case). The log term in the equation is also called the index of difficulty ( $ID$ ) and is usually measured in bits:

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

One of the limitations of the models that use only Fitts' law to model the movement time is that they assume that the movements are only restricted by human motor abilities. We will see later that in some cases the movement time is only a 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 a small or zero ID (e.g. key repeats). I will discuss this in more detail later on.

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. have determined the coefficients [Silfverberg et al. 2000] for the phone keypad identical to the one used in my experiments. The reported Fitts' law parameters were  $a = 176$  ms and  $b = 64$  ms/bit (Nokia 5100 series, thumb entry).

In some cases, as when *comparing* methods in which the letters are assigned to the same keys (e.g. *Multitap*, *T9*, *Less-Tap*), the inter-key movement time will be the same and can be assumed to be constant, without any sacrifice to the accuracy of the comparison.

### 3.2.2 Time for various key presses

Figure 15 below shows the times the keystrokes of different types – single, double, triple, quadruple etc. – take. I used the key log data from my experiments mentioned in Section 2. Note that it was obtained mainly from people who were not regular users of mobile phone text messaging (i.e. most of the participants were novices).

The times were measured from the last repeat of a key to the first press of a different key. There were also cases where the key was pressed more than four times (e.g. when trying to correct entry errors) but those cases were relatively rare. For simplicity, the graph shows average times for multiple key presses over all key digraphs, over all users and over both *Multitap* and *Less-Tap* methods. While the times for double through octuple key presses fit a linear relation very well into a line ( $R = 0.9975$ ), the difference between single and double keystrokes is noticeably larger. Also, note that the times are much greater than one would expect by considering Fitts' law alone. This discrepancy will be discussed further on in the chapter.

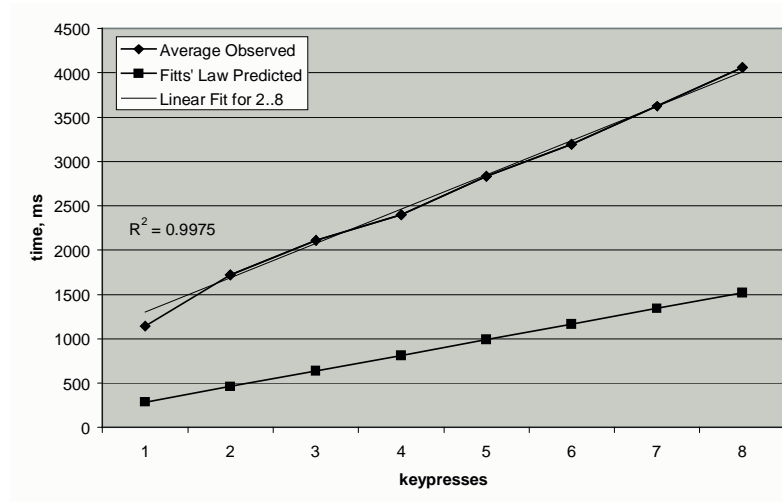


Figure 15: Time needed for multiple key presses (for non-expert users)

### 3.2.3 Key repeat time

By looking at the equation for Fitts' law, one would expect that the time to repeatedly press the same key would be equal to the coefficient  $a$  in the equation. It has been doubted for some time and, in fact, experiments have proven that Fitts' law should not be used blindly when the index of difficulty values are small [Soukoreff et al. 2002]. As an independent verification, I measured the key repeat time on a Nokia 5190 to be around 225 ms ( $\sigma = 21$ ) (this value is later referred to as  $T_{\text{repeat}}$ ). The number is an average computed over 10 people. Compare it to 176 ms as the  $a$  coefficient in the Fitts' law.

### 3.2.4 Mental overhead

Obviously, humans are not merely 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 one's limbs. Humans make mistakes (e.g. in the text entry task they hit a key different from the intended). Humans become tired – each session in my experiments was about 20 minutes long and, for some users, I could observe a noticeable drop in performance by the end of their sessions. Humans tend to make pauses to verify the progress they have made so far – to convince themselves that they haven't made an error *already* or that they will not make an error *in their next step*. These are factors that are possible to predict and quantify based on existing statistical data and to incorporate in a model. Some other factors are much harder to predict, for example, use of different strategies to perform the same task (e.g. in our case: focusing the visual attention on the

keypad, on the screen, or on the road). Such factors are not likely to be included in models due to their high variability.

From the key log data and my observations I identified the following time components, other than performing key presses:

1. Re-reading the phrase to be entered (most people prefer not to memorize the phrase they have to enter)
2. Figuring out which letter of which word has to be entered next (spelling out the word)
3. Determining which button should be pressed and how many times
4. Deciding if a second key press is required
5. If pressing a key more than twice, keep count of the number of presses made
6. [Verifying the result]

Of course, I cannot be 100% certain that the identified components are present for all users. However, I find it reasonable to believe that they are likely to be present in most, even in very experienced, users. Of course, there will be some common words, like “the”, “in”, “on”, which will be entered almost automatically. But other, more complex words will clearly require these components.

As it is hard to identify the components individually, since they directly follow one another, I will include only the combined times of several consecutive components (i.e. components 1, 2, 3, and 6 as the first, 4 as the second and 5 as the third coefficient) in my

model. The values of these coefficients are determined as differences between the average measured times to perform the keystrokes and the times that are predicted by Fitts' model alone.

Based on the data collected for *Multitap* and *Less-Tap* in my experiments (Section 2), each initial keystroke is preceded by about 906 ms ( $\sigma = 258$ ) of cognitive delay ( $D_{\text{init}}$ ), computed as the average observed movement time minus the average time predicted by Fitts' law. The first keystroke which falls onto the same key as the previous one is preceded by a 323 ms ( $\sigma = 131$ ) delay ( $D_{\text{repeat}}$ ), and each key repeat after that is preceded by about 125 ms ( $\sigma = 143$ ) ( $D_{\text{count}}$ ).

### 3.2.5 Time to enter a character (Movement Model)

From the above, I could derive the times required to enter a character using different text entry systems. I chose not to consider the Two-Key input method and some others that use non-standard keypads.

#### 3.2.5.1 Multi-press Input Methods

Multi-press Input Methods include *Multitap* and *Less-Tap*. For these methods, the time to enter a character modeled as:

$$T_{\text{char}} = D_{\text{init}} + T_{\text{Fitts}} + N_1 \cdot (D_{\text{repeat}} + T_{\text{repeat}}) + N_2 \cdot (D_{\text{count}} + T_{\text{repeat}}) + T_{\text{timeout}}$$

$T_{\text{Fitts}}$  is the time needed to move the finger from the preceding key to the current key, as computed from the Fitts' law equation.  $N_1$  is the number of "second" presses (either present – one or absent – zero).  $N_2$  is the number of key presses after the second press

(e.g. one for triple, two for quadruple etc.)  $T_{\text{timeout}}$  is the time that the user will have to wait for if the current character is located on the same button as the previous one. In my experiments the timeout kill key was not used, but the model can be easily modified to account for that, if desired (the value for  $T_{\text{timeout}}$  that I used was 1000 ms).

Obviously, with practice, the coefficients decrease in magnitude. I used the results for *Multitap* in [MacKenzie et al. 2001], where the authors analyzed *Multitap* and *LetterWise* in a longitudinal study (20 sessions, approximately 30 minutes each). Their study was different from the one described in Section 2 in that they used a large desktop-size numeric keypad. For simplicity and due to lack of the detailed data, I assumed that all three coefficients changed by the same relative amount. The following graph demonstrates my estimate and the power-law extrapolation for 20 more sessions. Please note that even at the 40<sup>th</sup> session, the delay is still almost 200 ms.

By varying the coefficient values, the model can be applied to model the performance of users of various levels of expertise. There is no single coefficient for “expertise” in the model, as the learning curves are different for different methods and depend on parameters such as complexity of the technique, complexity of the text that is entered, users’ previous experience etc.

Note that previous models also cannot completely account for various levels of expertise. The keystroke level model ([Card et al. 1980], section 3.1.1.1), can only model that motor performance improves (i.e. a  $T_k$  coefficient) and assumes that the mental preparation time stays the same over time. However, [Card et al. 1983] contains a table of suggested values for  $T_k$  for different levels of expertise and different tasks.

A model based on Fitts' law [Silfverberg et al. 2000], section 3.1.1.2) applies only to expert users. In various ways, people have repeatedly suggested that changing the coefficients  $a$  and  $b$  in the Fitts' law expression may be used to account for different levels of expertise. This approach, however, is fundamentally wrong, as Fitts' law was designed and verified to predict only the *motor* performance of humans; it clearly does not apply to cognitive issues, such as learning and recalling associations of letters to keys.

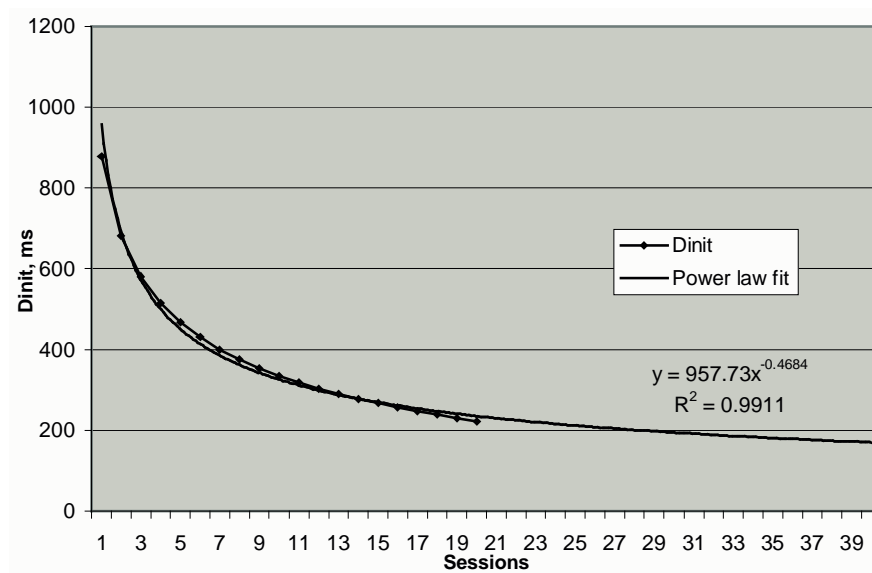


Figure 16: Coefficient  $D_{init}$  as a function of time

### 3.2.5.2 Predictive Input Methods

Predictive Input Methods include *T9*, *iTap* and several others. There are various possible strategies for such methods [James et al. 2001, MacKenzie et al. 2001, Silfverberg et al. 2000]. Several researchers mention that users often ignore the display until they finish

entering the word [James et al. 2001, MacKenzie et al. 2001]. Statistically, presses of the NEXT key account for less than 1% of the total number of key presses [MacKenzie 2002]. However, this does not mean that the users will not spend time verifying their input. Due to the difficulty of capturing this complex behavior of the users, I assume that that verification time is included in the time to prepare for the next key press ( $D_{\text{init}}$ ) in my model. I do this, since it is likely that the time to verify a proper word is not much different from the time to verify a single character, as in multi-press input methods. For simplicity, I also assume that the disambiguation is perfect. Both simplifications will cause the model to overestimate the speed. However, I believe that the degree of the overestimation will be small.

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

$$T_{\text{char}} = D_{\text{init}} + T_{\text{Fitts}}$$

I did not have as much experimental data on predictive methods as I did for *Multitap* and *Less-Tap*, but I find it reasonable to suspect that the keystrokes in those systems will be preceded by roughly the same amounts of cognitive delay as in the latter systems.

### 3.2.6 Linguistic Model

A linguistic model contains information about the frequency of different letter-pairs (digraphs). Like the design described in Section 2, my model is based on the letter-pair data from the British National corpus. The model is represented by a  $27 \times 27$  matrix, the 27 characters being 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 the sum of all cells equals to one.

Although the model is based on English only, adapting it to another alphabet-based language is as easy as analyzing a corpus and filling a similar square matrix with the newly computed probabilities.

### 3.2.7 Combining the Models

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

$$T_{\text{char\_in\_corpus}} = \sum \sum (p_{ij} \cdot T_{\text{char\_ij}})$$

where  $i$  and  $j$  range from 1 to 27 in our case (all possible letter pairs).

$T_{\text{char\_in\_corpus}}$  is the average time to enter a character in the corpus using a selected text entry method, in seconds.  $T_{\text{char\_ij}}$  is the average time to enter a character  $j$  after a character  $i$ , as computed in section 3.2.5.

To convert this number to words per minute (using the common assumption of 5 characters per word), I use the following expression:

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

### 3.3 Verifying the Model

#### 3.3.1 Model Predictions for Various Text Entry Methods

At this point, we can apply all the data mentioned previously to obtain the predictions of the new model. The computed results are in the following table:

<i>Technique</i>	<i>Speed (wpm)</i>
<i>Multitap</i>	6.97
<i>Less-Tap</i>	8.01
<i>T9</i>	10.07

Table 8: Predictions of the new model for various text entry methods

As mentioned previously, I used the Fitts' law coefficients from [Silfverberg et al. 2000] (the ones for the thumb) since my experiments involved a telephone handset of the same series. In my experiments the timeout key strategy was not used, since it was a short-term study (approximately one hour per method) and almost all of the users were non-experts. Based on my observations, the timeout kill key would be of limited value to most of the casual users of text messaging; and it was also noticed that even experienced users did not find a *short* timeout (1000 ms in the mentioned experiments) uncomfortable.

The predictions show that *Multitap* is the slowest, *Less-Tap* is a bit faster (by 15%), and *T9* is the fastest (26% faster than *Less-Tap* and 44% faster than *Multitap*). Note that the model is very generous towards *T9* by assuming that there is never a need to press the

NEXT key – if the intended word is not the most probable. In that, it is also implied that all of the entered words are in a built-in dictionary.

### 3.3.2 Comparison with Experimental Data

For convenience, all relevant data is shown in Table 9. This table shows results for novices.

<i>Technique</i>	<i>New Model</i>	<i>Model by Card et al.</i>	<i>Model by Silfverberg et al.</i>	<i>Results from James et al., Novice</i>	<i>Results from Section 2</i>
<i>Multitap</i>	6.97	18.35	22.3	7.98	7.15
<i>Less-Tap</i>	8.01	23.47	26.8		7.82
<i>T9</i>	10.07	24.97	40.6	9.09	

Table 9: Model predictions and experimental results for novices

In my experiments, the values obtained were 7.15 wpm for *Multitap* (2.6% more than predicted and 7.82 wpm for *Less-Tap* (2.4% less than predicted). The discrepancy here comes from the fact that in *Less-Tap* the key press rate was slightly lower (possibly due to presence of users who had some experience with *Multitap* and found *Less-Tap* to be interfering with their existing skills).

The study in [James et al. 2001] was slightly different as the authors of that paper had a relatively small set of phrases (8 phrases vs. 60). Thus, their numbers may be slightly larger due to less fatigue involved as well as due to some differences in the protocol (e.g. in my experiments, I did not ask the users to enter text “as fast as they could” which was

done mainly to reduce possible stress during long experiments). Yet, as one can see, the new model predicts the values observed in experiments much better. The only exception is the expert *T9* users (see Table 6). It is also possible that the set of words in the tested sentences was favourable to *T9*.

In the paper [MacKenzie et al. 2001] the authors try to investigate the issue of text entry in the predictive systems a bit further. Interested readers are encouraged to refer to the original text. It is known that about 1% additional keystrokes are required for *T9* [MacKenzie 2002, Silfverberg et al. 2000] but I could not determine the cognitive time preceding such relatively rare strokes.

### **3.4 Summary**

In this chapter, a new model was presented that can predict the text entry speed on standard 12-button telephone keypads. The major difference between this model and the existing models is that it includes mental overhead in its predictions in addition to movement time, and thus can quite accurately predict the performance of non-expert, as well as expert users. The values computed by the model (7 wpm for *Multitap*, 8 wpm for *Less-Tap*, and 10 wpm for *T9* for novice users) are reasonably consistent with those experimentally observed.

For *T9*, the model assumes perfect disambiguation since the analysis of the user-system interaction otherwise would be hard and extensive user tests would be required to gather enough key log data for the technique.

## Section 4

### Conclusions and Future Work

#### 4.1 Conclusions

In this thesis, I presented a new technique and a new model for text entry on 12-button telephone keypads.

The technique is easy to implement in existing devices, does not depend on a dictionary, supports eyes-free input, and is able to easily handle entry of text in different languages that use Latin alphabet. *Less-Tap* is very similar to the traditional technique used in phones, which is believed to be a beneficial characteristic, for there are indications that many other techniques – possibly fast and efficient – have failed to gain popularity only because they were too radical (e.g. theoretically Dvorak is faster than Qwerty [Bigler 2003], but it never caught on).

It has also been demonstrated that introducing *small* inconsistencies into a system (such as changing the ordering of letters on keys) is easily accepted by users and can, in many cases, bring a noticeable improvement. Some of the advantages of the new technique (e.g. the ability to mix different languages in the same sentence) may never be important for some users. Yet, I believe that having a choice among various text entry techniques is a good thing. Besides, *Less-Tap* can easily co-exist with other techniques based on the standard keypad layout.

Although the technique was designed to be optimal for English, it is very close to optimal for other languages as well.

In the model presented in the second part of this thesis, I tried to combine all the knowledge presently known about text entry modeling.

Previous models of text entry focused on predicting peak text entry speeds. A major distinctive feature of the new model, however, is the ability to quite accurately predict the performance of novice users. This ability is achieved through exploring the fact that different kinds of button presses are preceded by different amounts of cognitive delay. Previous models were either placing the same amount of delay before each press [Card et al. 1983] or were ignoring the delay completely [Silfverberg et al 2001].

## **4.2 Future work**

In this section, I will briefly explain how the work presented in this thesis can be extended.

### **4.2.1 Non-phone uses for the model**

Even though the model described in Chapter 3 applies to 12-button telephone keypads, there are some indications that it can be applied to other areas. For example, one can imagine entering capital letters on a QWERTY keyboard with a double key press, rather than through Shift or Caps Lock, as it is not common for a word to start with two

identical letters. Clearly, the approach used in the new model can also be used to model such technique.

#### 4.2.2 Measuring cognitive load

It is clear that different text entry methods may place different cognitive load on a user. In my experiments, very little emphasis was placed on determining the amount of mental processing involved. There are several ways to accomplish that. One is to have the users fill out a questionnaire designed specifically to measure the task workload, such as the NASA TLX task workload sheet [Hart et al 1988]. Another could be to have them perform a simple secondary task, for example a reaction time task. Directly measuring the brain activity could be more accurate, but it tends to be rather complicated (mainly due to interference between technical equipment and medical equipment).

#### 4.2.3 Further improvements to the model

It would be nice if there were a single parameter in the model, which would set the experience level. One of the ways to do that is by incorporating the power law of learning into the model – something attempted by Isokoski et al. [Isokoski et al. 2003].

The model described in this thesis was evaluated based on limited existing experimental data. In order to be able to evaluate the model more accurately, further empirical evaluations of the existing techniques would be useful, preferably comparing them to some common benchmark (such as *Multitap*, for example), and using the same corpus.

Multiple sessions would be necessary to take into account the learning effects shown in previous evaluations.

The research in the area of mobile input is by no means finished. During the last few years we saw an appearance of various mobile devices, such as phones, personal digital assistants (PDAs), MP3 music players, wristwatches running a PalmOS operating system etc. It is hard to tell what kind of devices we will encounter in the future. Yet, it is certain that there will be different types of devices that we cannot imagine at the moment. Some of these devices will require the development of completely new text entry techniques, which are adapted to the form factor and constraints of these devices. An adapted version of my model may then be useful to accurately predict the performance of the new text entry techniques on new devices.

## Bibliography

1. [BNC] British National Corpus, <ftp.itri.bton.ac.uk/bnc>.
2. [Bigler 2003] Jeff Bigler, *The Dvorak Keyboard*, 2003, [www.mit.edu:8001/people/jcb/Dvorak](http://www.mit.edu:8001/people/jcb/Dvorak).
3. [Card et al. 1980] Card, S. K., Moran, T. P. & Newell, A. The keystroke-level model for user performance time with interactive systems, *Communications of the ACM*, 23(7), 1980, 396–410.
4. [Card et al. 1983] Card, S.K., Moran, T.P., and Newell, A. *The psychology of human-computer interaction*. Hillsdale, NJ: Lawrence Erlbaum, 1983.
5. [Conklin 1987] Conklin, J. Hypertext: an introduction and survey, *IEEE Computer*, 1987, 17-41.
6. [dialABC] *Phone Keypads*, [www.dialabc.com/motion/keypads.html](http://www.dialabc.com/motion/keypads.html).
7. [Dunlop et al. 2000] Dunlop, M. D. & Crossan, A., Predictive text entry methods for mobile phones, *Personal Technologies*, 2000, 134–143.
8. [Eatoni] Eatoni Ergonomics, [www.eatoni.com](http://www.eatoni.com).
9. [Fitts 1954] 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.

10. [gsmworld] GSM World Association, [www.gsmworld.com](http://www.gsmworld.com).
11. [Gutowitz 2003] Gutowitz, H. Barriers to Adoption of Dictionary-Based Text-Entry Methods: A Field Study. To appear in the *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Budapest, Hungary, April 2003, available at [www.eatoni.com/research](http://www.eatoni.com/research).
12. [Hart et al. 1988] Hart, S. and Staveland, L., Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research, *Human Mental Workload*, P. Hancock and N. Meshkati, Eds. Amsterdam: North Holland B.V, 1988, 139-183.
13. [Hick 1952] Hick, W. E. On the rate of gain of information. *Quarterly Journal of Experimental Psychology*, 1952, 4, 11-26.
14. [Isokoski et al. 2003] 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*, New York: ACM, 2003, 752-753.
15. [James et al. 2001] 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*, New York: ACM, 2001, 365-371.
16. [Kober et al. 2001] Kober, H., Skepner, E., Jones, T., Gutowitz, H. & MacKenzie, I. S. *Linguistically optimized text entry on a mobile phone*, 2001, [www.eatoni.com](http://www.eatoni.com).

17. [Levy 2002] Levy, David. The Fastap Keypad and Pervasive Computing. *Proceedings of the First International Conference, Pervasive 2002*, LNCS 2414, Zürich, Switzerland, 2002, 58–68.
18. [Likert 1932] Likert, Rensis. A Technique for the Measurement of Attitudes, *Archives of Psychology*, No.140, 1932, p.55.
19. [MacKenzie 2002] 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*, Heidelberg, Germany: Springer-Verlag, 2002, 195–210.
20. [MacKenzie et al. 2001] MacKenzie, I. S., Kober, H., Smith, D., Jones, T., & Skepner, E. LetterWise: Prefix-based disambiguation for mobile text input. *Proceedings of the ACM Symposium on User Interface Software and Technology – UIST 2001*, New York: ACM, 2001, 111–120.
21. [MacKenzie et al. 2002] MacKenzie, I. S., Soukoreff, R. W. Text entry for mobile computing: Models and methods, theory and practice. *Human-Computer Interaction*, 17, 2002, 147–198.
22. [MacKenzie et al. 2003] 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*, New York: ACM, 2003, 754-755.

23. [Nesbat 2001] Nesbat, S. B. *Fast, Full Text Entry Using a Physical or virtual 12-Button Keypad*. Available at <http://www.exideas.com/ME/whitepaper.pdf>.
24. [Pavlovych et al. 2003] Pavlovych, A., Stuerzlinger, W. Less-Tap: A Fast and Easy-to-learn Text Input Technique for Phones. To appear in *Graphics Interface 2003*. Available at: <http://www.cs.yorku.ca/~wolfgang/publications.html>
25. [Pommerening 2003] Pommerening, Klaus (2003) *Cryptology*. University of Mainz, [www.uni-mainz.de/~pommeren/Kryptologie/English.html](http://www.uni-mainz.de/~pommeren/Kryptologie/English.html).
26. [Silfverberg et al. 2000] Silfverberg, M., MacKenzie, I. S., & Korhonen, P. (2000). Predicting text entry speeds on mobile phones. *Proceedings of the ACM Conference on Human Factors in Computing Systems – CHI 2000*, New York: ACM, 2000, 9–16.
27. [Soukoreff 2002] Soukoreff, R. W. *Text entry for mobile systems: Models, measures, and analyses for text entry research* (M.Sc. Thesis). Department of Computer Science, York University, 2002.
28. [Soukoreff et al. 1995] Soukoreff, R. W., MacKenzie, I. S. Theoretical upper and lower bounds on typing speed using a stylus and soft keyboard. *Behaviour & Information Technology*, 14(6), 1995, 370–379.
29. [Soukoreff et al. 2002] Soukoreff, R. W., MacKenzie, I. S., Using Fitts' Law to Model Key Repeat Time in Text Entry Models. Poster presented at *Graphics Interface 2002*.

30. [TrinColl 2002] *Cryptology Resources*, Trinity College, Hartford CT,  
<http://starbase.trincoll.edu/~crypto/resources/>.
31. [UND 2000] *The Crypto Drop Box*, University of North Dakota,  
<http://www.und.edu/org/crypto/crypto/stattools/ICs>.