

# Less-Tap: A Fast and Easy-to-learn Text Input Technique for Phones

Andriy Pavlovych

Wolfgang Stuerzlinger

Department of Computer Science  
York University  
Toronto, Ontario, Canada  
{andriyp, wolfgang}@cs.yorku.ca

## Abstract

A new technique to enter text using a mobile phone keypad, *Less-Tap*, is described. The traditional touch-tone phone keypad is ambiguous for text input because each button encodes 3 or 4 letters. As in *Multitap*, our method requires the user to press buttons repeatedly to get a required letter. However, in *Less-Tap*, letters are rearranged *within* each button according to their frequency. This way, the most common letters require only one key press.

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, after some initial training. For English, *Less-Tap* requires an average of 1.5266 keystrokes per character (vs. 2.0342 in *Multitap*).

We conducted a user study to compare *Less-Tap* against *Multitap*. Each participant had three 20-minute sessions with each technique. The mean entry speed was 9.5% higher with the new technique.

*Keywords:* Text input, mobile phones, communication, letter probabilities.

## 1 Introduction

Short text messaging (SMS) is very popular in the world. According to the GSM World Association, 24 billion messages were sent through GSM networks worldwide during May 2002. It should be noted, however, that the numbers are more modest for North America. Nevertheless, it is very likely that the number of text messaging users in North America will increase significantly in the near future.

A major problem in using text messaging, however, is how text can be entered on a telephone keypad.

### 1.1 Mobile Phone Keypad

The layout of the keypad of a telephone (Figure 1) is standardized. The standard (ITU E.161, also known as ANSI T1.703-1995/1999 or ISO/IEC 9995-8:1994) [8]

describes the layout of the 12 keys and the way the letters are assigned to keys.

### Mnemonic Numbers

The presence of letters on numeric keys is often used to create a mnemonic representation 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 assignment of letters to keys. However, it is not crucial to maintain the order of letters on each key.



Figure 1: The standard 12-key keypad.

### 1.2 Existing Approaches

Here, we will briefly describe the existing techniques for text entry for phone keypads.

#### Multitap

Most phones offer *Multitap* as the standard choice for text entry. In order to enter a letter with *Multitap*, a user presses 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 entering 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 using a dedicated key. The first approach requires fewer keystrokes; the

second tends to be faster for expert users, even though the number of key presses is greater [5].

### 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.

Another interesting two-key approach, *MessageEase* is described in [7]. However, it uses a non-standard layout. Both methods are not much faster than *Multitap* [9] and are not discussed further.

### Dictionary-based Disambiguation

There are several implementations 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. Once 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 this 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 means that text entry speed goes down if a desired word is one of the less likely choices in the dictionary.

This class of systems fails if the word is not in the dictionary or is misspelled. In this case, 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*.

There are additional indications that the actual process of entering text using dictionary based methods is somewhat distracting [2], main complaints being about the unpredictable nature of the system's response to the entry of non-dictionary words. 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 letters, this approach has a relatively high cognitive overhead, as the user needs to visually verify the result.

### Prefix-based Disambiguation

The only system that uses prefix-based disambiguation is *LetterWise* [5]. It 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 current 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.

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.

### Miniature QWERTY Keyboards

While the miniature keyboards seem to be a natural choice for the 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.

### Fastap [3]

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') and the letters are assigned to the buttons sequentially, in alphabetical order. To enter a letter, the user simply presses the corresponding key.

We are not aware of the authors' conducting a user study to compare the actual performance of their method against other alternatives.

## 2 Less-Tap

### 2.1 Motivation

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

### 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 [5]. This poses a major problem for dictionary-based disambiguation methods, as they limit the user to contents of the dictionary, more precisely to the dictionary built into the device. Words outside a 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 be not

supported by the devices that they can use. In addition, there are some nations in the world whose population is too small to warrant a 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 two cases, *Multitap* is currently the only option for text entry. The technique described in this paper, *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. For fairness, we must acknowledge that *LetterWise* also deals well with words from languages that were not considered when building its database.

### Eyes-free Input

Despite its modest speed, *Multitap* has an advantage over other existing systems. It allows experienced users, even those who have not reached the expert level yet, to enter text without looking at either the keypad or the display. This removes the visual focus of attention from the task and enables so-called eyes-free input [6]. Eyes-free input is very useful in certain situations such as working in confined environments, experiencing bad lighting conditions, exposure to excessive vibration, wearable computing, etc.<sup>1</sup> The new technique presented here, *Less-Tap*, is similar in that respect to *Multitap*, as the result of a key press is deterministic.

### 2.2 Letter Frequencies in English

The arrangement of letters on the keys in our method depends entirely on letter frequencies in English. In this paper, that data was derived from the British National Corpus [1].

### 2.3 Keystrokes per Character (KSPC) Metric

Keystrokes per character (*KSPC*) is a useful metric for comparisons between two text-entry methods [4]. For simplicity, we limit the following discussion only to lowercase letters and ignore capitals and special symbols. In this case, *KSPC* is equal to one for a standard ten-finger 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.

The published numbers are 1.1500 for *LetterWise*, 1.0072 for *T9*, and 2.0342 for *Multitap* with the use of a

<sup>1</sup> This in no way implies that we, the authors, condone the use of text messaging during driving, running, operating complex machinery and any other activity that requires full concentration.

timeout kill key [5]. *Less-Tap* with a timeout kill key has a *KSPC* value of 1.5266.

In our user tests, the timeout kill key was not used. That would give a *KSPC* value of 1.9488 for *Multitap* and 1.4412 for *Less-Tap*.

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. We will investigate this later in the paper.

### 2.4 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 keystroke, the second most frequent letter with two keystrokes and so on. Figure 2 illustrates the concept.

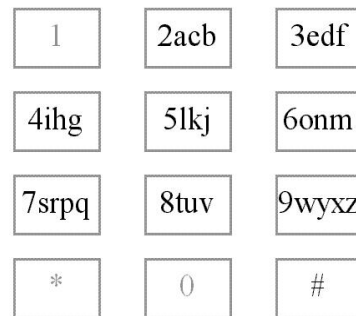


Figure 2: *Less-Tap* Layout.

By keeping the letters on the same keys, we ensure that the keypad remains standard compliant and that the 1-800 numbers can still be entered.

### The Distribution of Key Strokes

To illustrate the effect of our re-arrangement of the letters, we had computed the distribution of the number of key presses required to enter the complete text of our corpus.

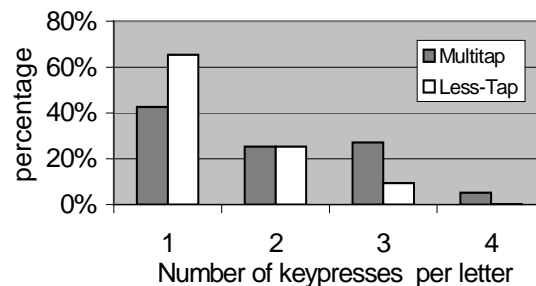


Figure 3: Frequency of different 'multistrokes'.

As shown in Figure 3, *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.

In the next section we will present an empirical evaluation of *Letter-Wise*. As in [5], we used *Multitap* as the technique to compare our technique against.

### 3 Test Method

#### 3.1 Participants

There were 12 participants in the test who were recruited through advertisements posted on the university campus. 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.

#### 3.2 Apparatus

##### Hardware

Unlike similar studies that used a desktop keypad [5], we used an actual Nokia 5190 handset. To implement our technique we connected the keypad of the mobile phone to the keyboard of a PC. Here we exploited the fact that both keyboards are based on a matrix layout, i.e. horizontal and vertical rows. We 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, we were able to maintain the form factor, weight, size, as well as the typical 'feel' of a mobile phone. The buttons were *not* re-labeled to reflect the different layouts of the letters. While re-labeling could increase the performance for the new method, we decided to keep the *hardware* modifications to the phone to a minimum. The 5100-series models are relatively common which should make it easy to replicate this study. In the experiment we used only the 12 standard keys as shown in Figure 2.

Since displaying the characters on an LCD display of the (non-working) handset is hard to achieve for technical reasons, we chose to display the entered text on a Wacom LCD tablet. The advantage of using a tablet was that it allows users to quickly find 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 chose to hold the handset on their laps or in the air.



Figure 4: Equipment used in the experiment.

##### Software

The software that we used in the experiment was written in Java and had been used previously for text entry experiments [5]. We modified it to interpret the characters entered on the connected phone keypad. Furthermore, we added the option to use either *Multitap* or *Less-tap* (referred to as Alternative Text Entry Method during the experiment). Figure 5 shows the user interface.

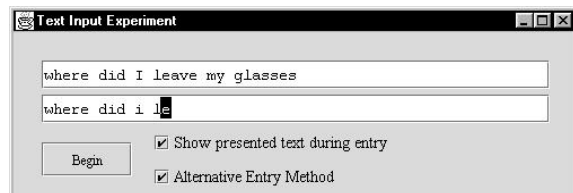


Figure 5: Snap shot of the program window.

The first text field shows the phrase to enter 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 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 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 '1' so that the participants could rest between the phrases at their discretion. They were informed about this feature of the system.

As we were testing first time users, we did not provide a timeout kill key in our design. This decision is based

on indications that the use of a timeout kill key has a noticeable advantage only for expert users [9]. 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 has expired or not. In *Multitap* and *Less-Tap*, pressing the same button within the timeout changes the last letter entered.

We also used a second version of the program in order to test our technique in "eyes-free" mode. In that version, the 2<sup>nd</sup> text field as in Figure 5 was completely blanked out (even the cursor was not visible).

### Set of Phrases

The set of phrases was the same as the one used in the test of LetterWise [5]. It is representative of English [5]. The phrases to be entered were chosen randomly from this file.

## 4 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. We used a within-subjects design and counterbalanced the order of entry methods to compensate for learning effects.

### Instructions

Before the test, the participants were given brief instruction as to their task, how the software worked, 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 appear 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 2). These sheets were normally placed at the bottom-left corner on the screen.

### 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, we did not force the participants to enter the phrases completely error-free. They were told that the errors could be corrected via using a backspace ('\*') button. Also, if

they pressed a correct key but a wrong number of times, they could continue pressing it until the letter appears for the second or third time.

There were occasional "uncorrectable" errors when users pressed an "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.

### Eyes-free sessions

In their 3<sup>rd</sup> 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.

## 5 Results

### 5.1 Entry Speed

Overall, the mean entry rate was 7.15 wpm for *Multitap* and 7.82 wpm for *Less-Tap*. In other words, *Less-tap* was faster by 9.5%. The main effect for entry method was statistically significant ( $F_{1,11} = 7.32, p < 0.05$ ). Figure 6 demonstrates the entry speed in words per minute for different sessions.

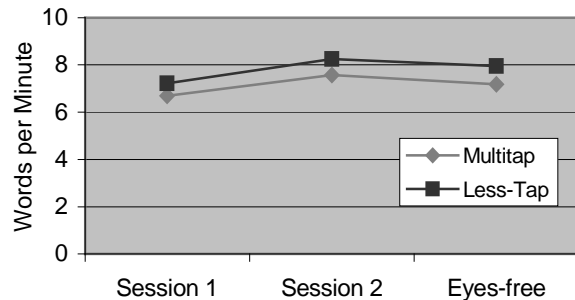


Figure 6: Entry speed (wpm) by entry method and session.

### 5.2 Error Rate

Over all sessions, the difference in error rate was not statistically significant between *Multitap* and *Less-Tap* ( $F_{1,11} = 4.66, p > 0.05$ ). However, it was significant for the eyes-free session ( $F_{1,11} = 4.91, p < 0.05$ ). Figure 7 shows the graphs for error rate for each session.

### 5.3 Errors: Unnecessary Key Presses

To analyze the errors further, we computed the number of unnecessary key presses. This 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 a backspace button, almost all users preferred to continue pressing the same button thus going for another round (e.g. the letter 'u' can be entered by pressing the button '8' two, five, eight or more times).

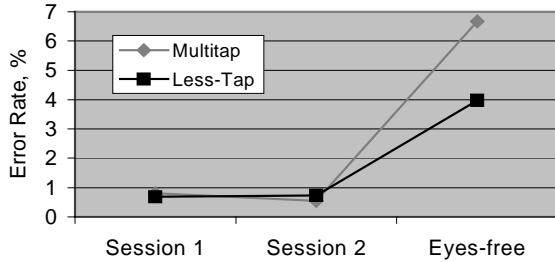


Figure 7: Error rate vs. entry method and session.

The data for following figure was obtained by analyzing the key press log data. The absolute number of errors 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 the results, we computed the percentage of extra key presses in the total number of key presses of each type. The resulting graph is shown in Figure 8. For example, if 2700 key presses were required to enter all letters that need two key presses each, and 3000 were actually performed, than the percentage of extra presses would be 10%.

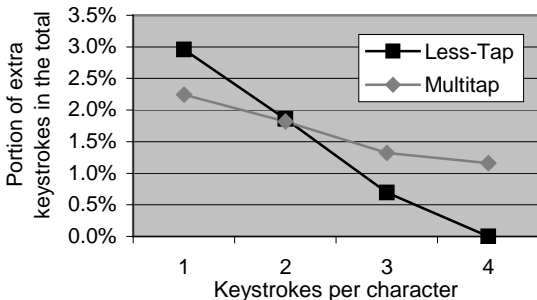


Figure 8: Total number of errors vs. total number of key presses of that type.

The graph shows that the number of errors is significantly greater for single key presses in *Less-Tap*. We 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 3).

#### 5.4 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 ( $F_{1,11} = 53.97, p < 0.05$ ). Figure 9 shows the graphs by session.

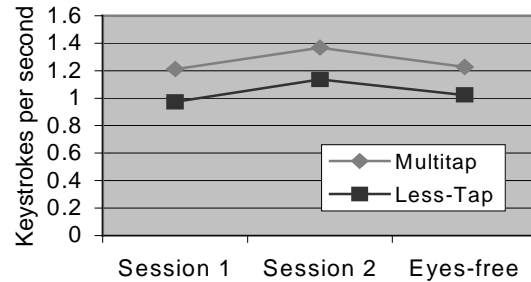


Figure 9: Key repeat rate in different sessions.

As far as we can determine, the fact that *Less-Tap* has always fewer keystrokes per second (KSPS) than *Multitap* can be attributed to cognitive overhead for unusual letter sequences (i.e. ACB instead of ABC).

#### 5.5 Keystrokes per Character

The number of keystrokes per character was highly significant between methods ( $F_{1,11} = 332.74, p < 0.001$ ).

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	1.9488	1.4412

Table 1. KSPC data by session.

For both methods, the values were 10–15% greater than those predicted. Overall, *Multitap* required about 33% more keystrokes, which (almost) matches the prediction (35%).

#### 5.6 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. For example, the choices for perceived speed were: 'much faster', 'faster', 'slightly faster', 'no difference', 'slightly slower', 'slower', 'much slower'.

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

- faster
- slightly easier to use
- slightly less tiring
- slightly easier to learn
- somewhat preferred, in that they wanted *Less-Tap* as a permanent replacement of *Multitap* in their phone.

Most commented that they would prefer *Less-Tap* more if the buttons were re-labeled to reflect the changed ordering.

## 6 Discussion

### 6.1 Entry Speed vs. Sessions

It can be seen that the speed improves as the time goes on. That can be seen from both the improvement of key repeat time (Figure 9) and the improvement of text entry speed in WPM (Figure 6).

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, we asked participants to work in one-hour slots. The analysis of the log files indicated that by the end of their respective time slot, the speed dropped for some of the users.

### 6.2 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 the text entry speed is only 9.5%?

The speed of the text entry in words per minute is composed of 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 character. In this paper, ‘word’ is defined as five characters, a convention commonly used in the analysis of text entry methods.

The results above show that the discrepancy between the actual and predicted values for KSPC is similar for both techniques. Now, let us go back to the Figure 9. As we can observe, the key repeat rate is significantly lower for *Multitap* than it is for *Less-Tap*. It took about 20% longer to perform a button press in *Less-Tap*. We attribute some of this difference to *cognitive overhead*.

#### Cognitive Overhead

Cognitive overhead 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.

While we expected *Less-Tap* to be about as easy to learn as *Multitap*, obviously there were some differences which caused the users to be slower in learning *Less-Tap* or using it (even though they thought the opposite and actually showed higher entry rates). We attribute this to additional mental processing required by *Less-Tap*. Considering that, overall, participants found *Less-Tap* “easier to use” and to be “slightly less tiring”, we believe that the increased cognitive overhead caused less fatigue than more button presses.

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

It would have been interesting to compare the findings mentioned above to those of alternative systems, notably, *LetterWise* [5]. While there is no data that states a key repeat rate for *LetterWise*, there is a very detailed graph [5, 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. We estimated from their graph the values for the 3<sup>rd</sup> 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 50% 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 9), we may conclude that the cognitive overhead of our system is smaller than that of *LetterWise*. We could find no similar data on *T9*.

### 6.3 Presence of Two Distinct Groups of Users

When analyzing the results we discovered that there seem 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% (8.54 wpm vs. 7.12 wpm), with high statistical significance ( $F_{1,5} = 83.87, p < 0.001$ ). In the other group, the two methods were essentially the same in terms of entry speed and error rate (for entry speed,  $F_{1,5} = 0.13, p \gg 0.05$ ). In the second group, three users performed slower with *Less-Tap* but only for one of them was the difference greater than 2% and in that case it was caused by a poor performance of *Less-Tap* during the “eyes-free” session.

We speculate 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, we believe that with more use eventually all people will adapt to the new system and benefit from it. A long-term study would be needed to analyze this further.

### 6.4 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 7). Of course, some more practice is required in order for the error rate to drop to a tolerable level.

## 6.5 Comparison with Other Techniques

There are several methods to enter text on the mobile phone keypad. None of them is perfect, in our opinion (including our own). In the following table, we will try to briefly summarize the advantages and disadvantages of different techniques.

Method	Multi-tap	Less-Tap	Letter-Wise	T9
Property				
Available on mobile phones	Almost all	None	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	Must change dictionary

\* – includes memory and processing requirements

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

Table 2. Comparison between different text entry methods.

## 6.6 Future Directions

In this paper, we did not test our technique against another popular technique – T9. Since the two methods work very differently, it's hard to speculate how *Less-Tap* would compare to T9 in the real life.

Also, it should be noted that the speed at which key presses were made was significantly lower than theoretically possible (see [9], for example) – in both systems. Thus, it is likely that, with further training, the advantage of *Less-Tap* over *Multitap* will increase.

## 7 Conclusion

We presented a new technique to enter text on mobile phone keypads, which requires 25% fewer keystrokes compared to *Multitap*. The results of the user study show that for first-time users the new technique is on average

more than 9.5% faster. In several cases we were able to observe an over 30% speedup.

*Less-Tap* is very easy to implement, like *Multitap*. It also allows for eyes-free input and does not depend on a dictionary.

## Acknowledgements

The authors would like to thank Scott MacKenzie for providing us with software for the experiments and William Soukoreff for his valuable insights on how to conduct user studies, as well as NSERC for funding.

## References

- [1] British National Corpus, <ftp://ftp.itri.bton.ac.uk/bnc>.
- [2] Gutowitz, H. Barriers to Adoption of Dictionary-Based Text-Entry Methods: A Field Study. Available at <http://www.eatoni.com/research>.
- [3] Levy, David. The Fastap Keypad and Pervasive Computing. In *Proc. First International Conference, Pervasive 2002*, LNCS 2414, pp. 58-68, Zürich, Switzerland, 2002.
- [4] MacKenzie, I. S. KSPC (keystrokes per character) as a characteristic of text entry techniques. To appear in *Proc. Fourth International Symp. on Human-Computer Interaction with Mobile Devices*. Heidelberg, Germany: Springer-Verlag.
- [5] MacKenzie, I. S., Kober, H., Smith, D., Jones, T., & Skepner, E. (2001). LetterWise: Prefix-based disambiguation for mobile text input. *Proc. ACM Symp. on User Interface Software and Technology – UIST 2001*, pp. 111-120. New York: ACM.
- [6] MacKenzie, I. S., & Soukoreff, R. W. Text entry for mobile computing: Models and methods, theory and practice. To appear in *Human-Computer Interaction*.
- [7] Nesbat, S. B. Fast, Full Text Entry Using a Physical or virtual 12-Button Keypad. Available at <http://www.exideas.com/ME/whitepaper.pdf>.
- [8] Phone keypads. <http://www.dialabc.com/motion/keypads.html>
- [9] Silfverberg, M., MacKenzie, I. S., & Korhonen, P. (2000). Predicting text entry speeds on mobile phones. *Proc. ACM Conference on Human Factors in Computing Systems – CHI 2000*, pp. 9-16. New York: ACM.