Mobile Text Entry

Amal Sirisena
Department of Computer Science
University of Canterbury
Christchurch, New Zealand
ans26@cosc.canterbury.ac.nz

Supervisor: Andy Cockburn

November 8, 2002

Abstract

There has been a substantial growth in interest in mobile text entry over recent years, among both researchers and users. Increasingly mobile devices are being used to perform text-intensive applications, such as text messaging, creating a demand for more efficient and easier to use text entry methods. Unlike for desktop computing, no single, standard mobile text entry method has emerged. The diversity of mobile devices makes it unlikely that this will ever occur. Thus, mobile text entry remains a very open area of research, providing a favourable environment for the development of innovative text entry methods. A necessary part of the development of a new mobile text entry method is a comparison of its performance with existing methods. Despite being complex and time consuming, empirical evaluations remain the best way to make these comparisons.

A review of current best practice for the empirical evaluation of mobile text entry methods is presented, alongside a classification of existing mobile text entry methods. The results of an empirical evaluation of a new mobile phone text entry method called Fastap are reported. The performance of the new method, along with that of the T9 and multi-press with timeout mobile text entry methods, was measured for the entry of four different types of text and with three different levels of user experience. The Fastap method was found to provide the best immediate usability among the three methods and its performance continued to improve as users gained more experience with it. Fastap also performed strongly in the subjective ratings. The results of the evaluation are very positive for the ongoing development of the Fastap interface.

Keywords: Mobile Devices, Text Entry, Fastap

Contents

Chapter 1

Introduction

The growth in the global use of Short Message Service (SMS) text messages has shown no signs of abating. The GSM Association reports that, worldwide, approximately 24 billion SMS text messages were sent in May, 2002. The forecast total number of SMS text messages for the whole of 2002 is 360 billion, a 44% increase on the 250 billion SMS text messages sent in 2001. Yet, text messaging has flourished in spite of, rather than because of, the support provided for it in most mobile phones (James & Reischel 2001). The affordances provided by other mobile devices, such as Personal Digital Assistants (PDAs), are just as poor.

Text messaging is only one of the many factors that is driving demand for fast, efficient and easy to use mobile text entry methods. The extension of text-intensive applications, such as web browsing and word processing, to mobile devices has highlighted the inadequacies of the current mobile text entry methods. Increasingly, mobile devices are becoming consumer products rather than technical tools. This new class of users expect the mobile devices to provide the same level of usability and efficiency as other consumer devices.

Text entry research for desktop computers has been focused on niche or futuristic applications because of the unchallenged dominance of the Qwerty keyboard for general purpose text entry. No such standard, general purpose text entry method exists for mobile devices. The development of such of a method is unlikely given the wide range of mobile devices in use, with each type of device placing its own requirements and restrictions on potential text entry methods.

In response to the demand for better mobile text entry methods, numerous different methods have been developed. Each method must make two fundamental tradeoffs: between potential efficiency and training time and device size and character-set size (Gopher & Raij 1988). The best way to measure how well a given method manages these tradeoff is through empirical evaluations. Of most interest is the comparison between new mobile text entry methods and current best practice. Designing an empirical evaluation that generates valid and repeatable results is not a trivial task.

Chapter 2 presents a review of previous work in mobile text entry, in particular the empirical evaluation of mobile text entry methods. A classification of a wide range of mobile text entry methods is also presented. Chapter 3 describes a comparative evaluation of a new mobile phone text entry method called Fastap with the two most widely used mobile phone text entry methods, T9 and multi-press. Chapter 4 discusses potential avenues for further work. The final chapter presents the conclusions of the report.

Chapter 2

Background Work

Designing new text entry methods for mobile devices is expensive and labour-intensive (Silfverberg et al. 2000). The assessment and comparison of a new text entry method with current methods is a necessary part of the design process. The best way to do this is through an empricial evaluation. Unfortunately, such evaluations are time-consuming and complicated. Careful planning and execution is required when undertaking such an evaluation, as an abundance of confounding factors exist that could negatively effect its repeatability and validity.

To control confounding factors, empirical evaluations place participants in constrained, artificial environments. This allows the behaviour or behaviours of interest to be isolated and thus accurately measured. To ensure the validity of an evaluation, however, it has to be designed to be as representative of actual user behaviour as possible (MacKenzie & Soukoreff 2002). MacKenzie & Soukoreff suggest this need not result in a trade-off between accuracy and relevancy. Instead, evaluations should be designed to maximise both relevancy and accuracy.

2.1 Empirical Evaluation of Mobile Text Entry Methods

2.1.1 Evaluation Procedure

A basic difference between evaluations and actual usage is in the tasks participants are asked to perform. In real life, mobile text entry tasks are text creation tasks, with the user generating the text at the time of entry. Empirical evaluations, however, have tended to use text copy tasks (Butts 2001, MacKenzie & Zhang 1999, MacKenzie & Zhang 2001, Dunlop & Crossan 2000, MacKenzie, Kober, Smith, Jones & Skepner 2001, James & Reischel 2001). With text copy tasks, participants enter a pre-prepared text. MacKenzie & Soukoreff cite several important reasons (control over content, ease of error identification and not introducing confounding behaviours) for why text copy tasks have been favoured over text creation tasks, despite text creation tasks being more representative of real user behaviour. Hybrid approaches, such as having participants memorise short, easy to remember phrases before each task or interleaving input and output, capture the advantages of both text copy and text creation tasks (MacKenzie & Soukoreff 2002).

Usability, as measured by both qualitative and quantitative techniques, is highly dependent on a user's level of experience with an interface. For the purposes of empirical evaluations the two stages of user expertise of most interest, novice and expert performance, can be the most difficult to measure (MacKenzie & Soukoreff 2002). The immediate usability of an interface for novice users is important because their initial reaction will determine their willingness to invest the time and effort required to master it (Shneiderman 1998). To truly measure immediate usability, the participants' exposure to the interface must be carefully managed (MacKenzie & Zhang 2001) and perhaps their previous experience scrutinised. Measuring expert performance presents logistical problems, as it requires that numerous experimental sessions be conducted over a relatively long period of time. These longitudinal studies are necessary to establish a learning curve for a particular interface and to gauge its optimal performance.

Past evaluations have focused on the collection of quantitative measures, in particular the measurement

of text entry speed (words per minute) and accuracy (error rates), rather than qualitative ones (James & Reischel 2001, MacKenzie et al. 1999). Proper interpretation of quantitative measures requires consideration of the participants' subjective responses to an interface, which can only be captured by qualitative tests. Statistical methods are available to analyse the non-parametric data generated by qualitative measures, but the reluctance of experimenters to use these measures may be explained by the time and effort that must be expended to create appropriate tests. One solution to this is to use standardised qualitative tests, which reduces the workload of experiment designers and also improves the repeatability of the experiment. An example of a standard subjective workload measurement tool is the NASA Task Load Index (TLX) (Hart & Staveland 1988).

Another issue to consider when designing an empirical evaluation is the apparatus that will be used. Ideally, the devices used in evaluations should be the same as those used in the real world. There are many reasons, however, why this may not possible. For new text entry methods, a working prototype may not exist and the resources to build one may not be available (Silfverberg et al. 2000). Even where actual implementations are available, they may not be appropriate. The primary reason for this is that mobile devices, especially mobile phones, are not easily, if at all, modifiable. This makes it difficult for experimenters to exert as high a degree of control over experimental conditions as they would like. An alternative approach is to use software-based emulators (Butts 2001, Dunlop & Crossan 2000, MacKenzie, Kober, Smith, Jones & Skepner 2001, MacKenzie & Zhang 1999). These allow experimenters to automate most aspects of the evaluation and give them total control over the conditions under which the evaluation is run. The cost of this approach is a reduction the in the validity of experiment, as how participants interact with the emulators is different from how participants interact with actual mobile devices. Using actual devices does have some drawbacks. As noted above, the control experimenters can exert over experimental conditions is reduced. Another drawback is a reduction in which aspects of the evaluation can be automated. In particular, data collection must be performed manually, adding to the time and effort invested in conducting the evaluation. The overriding benefit of this approach is a significant improvement in the validity of the evaluation's results.

2.1.2 Dependent Measures

Text entry speed and accuracy are the two standard dependent variables used in mobile text entry evaluations (MacKenzie & Soukoreff 2002).

The metrics for measuring text entry speed are well established. The simplest measure of the text entry speed is characters per second (CPS), which is calculated as:

$$\frac{C_n}{T_c} \tag{2.1}$$

where C_n is the number of characters entered and T_c is the time, in seconds, taken to enter those characters. One pitfall with measuring task completion time is the points at which timing is started and stopped (Butts 2001). If the timing starts on entry of the first character, which is often the case with automated evaluation software, the mental and physical preparation time for entering the first character is not included in the final task completion time. In this case, the character count should be reduced by one when calculating CPS (Butts 2001). Timing should be terminated upon entry of the last character in the source text.

Typically, text entry speeds for typing on desktop computers is measured in words per minute (WPM). This metric has also become the standard for reporting mobile text entry speeds and can be derived from characters per second (2.1) by:

$$CPS \times \frac{60}{W_c} \tag{2.2}$$

where 60 is the number of seconds in a minute and W_c is the average number of characters in a word. A word in this context can include any possible input character, not just letters. By convention, the number of characters in a word is set to five, which is the standard typist's definition MacKenzie & Soukoreff. Different values have been used for this parameter, based on the analysis of a corpus of representative text. The most widely used alternative value is 5.98 (Dunlop & Crossan 2000, James & Reischel 2001). One

pitfall with basing calculations of word length on standard corpora is that they may not be representative or statistically consistent with the text used in mobile text entry (MacKenzie & Soukoreff 2002). A corpus created from the actual text (especially Short Message Sevice (SMS) text messages) entered into mobile devices would be a superior basis for language model calculations (Dunlop & Crossan 2000). Creation of such a corpus would be difficult, because mobile network operators may be unwilling to release details of the messages sent over their network due to privacy and commercial sensitivity concerns.

Unlike text entry speed, text entry accuracy is more problematic to measure. At a practical level this can be attributed to the difficulty of automating error measurement and analysis, while at the theoretical level the underlying cause is the compounding nature of errors (MacKenzie & Soukoreff 2002). Both the type of errors that occur and their cause are of interest. Four basic error types have been identified: character substitution, character omission, spurious character insertion and character transposition (MacKenzie & Soukoreff 2002). Errors can be traced back to aspects of an interface's functionality. For example, a common cause of character substitution errors that occur when using the multi-press mobile text entry interface is pressing the key on which the target character appears too many or too few times (see Section 2.3.1).

The difficulty of error analysis often leads experimenters to ignore errors completely, perform only informal error analysis or artificially constrain tasks to prevent errors (MacKenzie & Soukoreff 2001). None of these approaches is ideal. Understanding the trade-off between speed and accuracy is necessary to to able to accurately characterise the performance of a text entry method, by finding its net text entry speed. A more realistic approach to analysing accuracy is to instruct participants to "correct as they go" (MacKenzie & Soukoreff 2002). This approach leads to two classes of errors: those that were corrected and those that were not (MacKenzie & Soukoreff 2001). MacKenzie & Soukoreff (2002) identify a potential problem with this approach. If participants do not immediately identify that an error has occurred then they may enter a significant amount of additional text before noticing the previous error. In this case, the overhead involved in correcting the errors will be substantial (Matias et al. 1996). This is particularly problematic for the evaluation of key-based input methods that do not require constant eye focus on the interface display.

A metric for measuring the overhead involved in correcting errors is keystrokes per character (KSPC) (MacKenzie 2002). KSPC, for a text entry task, is calculated as:

$$\frac{K_n}{C_n} \tag{2.3}$$

where K_n is the number of keypresses made and C_n is the number of characters entered. This measure is appropriate for interfaces that have an error-free KSPC of close to 1.0, such as a QWERTY keyboard (MacKenzie 2002). Error correction requires additional keypresses over and above those required for error-free text entry and this overhead is reflected in the calculated KSPC value. This measure is less appropriate for interfaces where the average KSPC is significantly higher than 1.0. For example, the average KSPC for the multi-press with next method (see Section 2.3.1) has been calculated as 2.0342 (MacKenzie 2002). An alternative approach is find the ratio of actual keypresses made to the minimum keypresses required. This can be calculated using the formula:

$$\frac{K_n}{K_{min}} \tag{2.4}$$

where K_n is the numbers of keypresses made and K_{min} is the minimum number of keystrokes required to complete the task. This approach has several advantages over KSPC. For any interface, keystroke minimising, error-free text entry will result in a ratio of 1.0. Using the same baseline for any interface allows comparisons between different interfaces (with varying average KSPCs) to be made on a more valid basis. Calculating the value of K_{min} for a given source text and interface should not be difficult; however, it should be noted that minimising the number of keystrokes made is not the same as optimal use (MacKenzie 2002). For example, with the T9 predictive text entry interface, when entering non-dictionary words users have the choice of entering a word into the built-in dictionary or switching to multi-press mode to enter the word (see Section 2.3.1). The former strategy requires more keypresses but would be better in the long run, because after words are entered into the dictionary they can be subsequently provided as predictions. For the purposes of calculating K_{min} the latter strategy would be assumed. Another problem with this method is that the additional keystrokes, as reflected in the calculated ratio, may be attributable to sub-optimal use of the interface, not just error correction.

To measure uncorrected errors requires a comparison of the source and transcribed texts. The error rate can be calculated by determining the minimum string distance (MSD) between the two strings, defined in terms of the editing primitives insertion, deletion and substitution (MacKenzie & Soukoreff 2001).The MSD is the smallest set of the editing primitives that when applied to the transcribed text produce the source text. MacKenzie presents both an algorithm for calculating the MSD statistic for two strings and the following for finding the error rate:

$$ErrorRate = \frac{MSD(A,B)}{max(|A|,|B|)} \times 100\% \tag{2.5}$$

where A is the source text, B is the transcribed text and $|A|$ and $|B|$ are the length, in characters of the source and transcribed text, respectively.

2.2 Alternatives to Empirical Evaluations

2.2.1 Contextual Enquiries

Contextual enquiries or field studies, such as the one conducted by (Grinter & Eldridge 2001), are necessary to understand the tasks that users want and need to perform with an interface, the context in which the interface is used and what skills users bring to their use of an interface. For example, Grinter & Eldridge found their participants made heavy use of abbreviations (146 uniques ones were logged) to speed up text entry. The abbreviations used combined letters, numbers and punctuation. This suggests any evaluation should test the efficiency of entering these abbreviations and not just traditional text. Information such as this, gleaned from contextual studies, is invaluable not only for the development of new mobile text entry methods but also for designing evaluations that will produce relevant results.

2.2.2 Predictive Models

An alternative to empirical evaluations is to develop models that predict the performance of text entry methods (Butts 2001, Silfverberg et al. 2000, Dunlop & Crossan 2000). The models aim to allow designers and researchers to easily evaluate alternative text input methods without having to invest significant resources in implementations and empirical studies. An example of a problem that predictive models can overcome is associated with comparitive evaluations of new text entry methods with well-established, widely used alternatives (Moyle 2001). In these situations it is difficult to find users who are complete novices with the existing text entry methods. To overcome this requires that users of the new text entry method receive enough training to make them expert users as well, which could require a significant amount of time.

The two main approaches to developing predictive models have been keystroke-level modelling (KLM) (Card et al. 1980) and movement time prediction (MacKenzie & Soukoreff 2002). The models are primarily used to predict expert, error-free performance, which allows simplifying assumptions to be made. For example, in their predictive models for expert text entry rates with different text entry interfaces, Silfverberg et al. did not incorporate the time required to visually scan the keyboard to find each character, reasoning that expert users would be entirely familiar with the keypad layout. A model for predicting non-expert text entry would have to include this component.

If a model will be used to predict text entry speeds it must incorporate a language model (MacKenzie & Soukoreff 2002). These models are created by analysing a representative corpus to establish the relative frequency of character combinations (individually and in pairs or triples), words or phrases in a language of interest. Care must be taken to ensure that the corpus upon which the analysis is performed is representative of the user language and reflects both input modalities and the editing process used (MacKenzie & Soukoreff 2002).

Keystroke-level models break down users' interactions with an interface into a set of low level tasks, each task with an estimated completion time. Tasks at this level of decomposition do not require decision-making or deep-thought. For example, Dunlop & Crossan proposed a keystroke level model composed of the time taken to press a key, move the hand to the keyboard and mentally prepare to press a key. Combining these keystroke-level time estimates with the average KSPC and word lengths in characters (derived from

text analysis) allowed the prediction of text entry rates. The time estimates associated with these tasks were taken from the original KLM evaluation.

Butts took a different approach to creating a predictive model by basing it on average user behaviour rather than expert behaviour. The model was based on data gathered from a pilot study, as well text analysis. The model included the time to press a key (an empirically derived constant), the average KSPC (which varied between interfaces) and the average number of characters per word (set to 5). The predictions generated from this model were compared to the results of a subsequent comparative evaluation of the multi-press with next and T9 predictive text entry interfaces. While the predictions for multi-press with next were within 10% of actual performance, the predcitions for T9 were less accurate, though better than those generated by more complicated models (Silfverberg et al. 2000, Dunlop & Crossan 2000).

Movement time prediction models are constructed from estimates of the time required for each of the movements users make when interacting with an interface. Fitts' Law (Fitts 1954) is used to generate the time estimates for movements such as moving to a key or pressing a key. These times are used to calculate the time required to enter each character from the language of interest. To predict text entry rates, these predicted character entry times are combined with the letter-pair probabilities to derive the average character entry time. The reciprocal of the formula can then replace the characters per second component of the word per minute formula (see Equation 2.2).

The combinatorial optimisation of different aspects of text entry methods, such as identifying optimal (movement minimising) key layouts for soft keyboards (see Section 2.3.2) (Zhai et al. 2000, MacKenzie & Zhang 1999), has been another area where modelling has been applied. The complexity of such optimisation means trial-and-error or brute-force approaches are not possbile (Zhai et al. 2000). A more fruitful approach has been to use the approaches, such as simulated annealing, used to solve combinatorial optimisation problems in other areas to find potentially optimal solutions (Mankoff & Abowd 1998).

The two types of predictive models described can be created for both key-based and stylus-based text entry methods. To model stylus-based methods using keystroke-level models, however, the text entry process used should be equivalent to pressing a key. For example, the tapping of a stylus on a soft keyboard could be modelled using a keystroke-level model but handwriting recognition could not.

While predictive models have been used successfully with many different types of mobile text entry methods, predictive models for text methods based around the telephone keypad have been shown to be highly inaccurate when compared to the results of actual evaluations (Butts 2001, James & Reischel 2001, Dunlop & Crossan 2000, Silfverberg et al. 2000). A common failing among all models is their inability to properly measure the cognitive effort required when using a given text entry method. Unfortunately, it is difficult to determine cognitive effort empirically. Due to this problem and others, it has not yet been possible to develop a generic predictive model that can be used to evaluate any text entry method. Thus, empirical evaluations, despite their drawbacks, remain the best way to evaluate text entry methods.

2.3 Classification of Mobile Text Entry Methods

Mobile text entry methods have tended to develop through evolution rather than revolution (Isokoski 1999). Deficiencies in existing text entry method have often served as the motivation for the development of new methods, which while improving upon the original method still retain significant aspects of its design. Some examples of this relationship are multi-press and one-key with disambiguation (see Section 2.3.1), Unistrokes and Graffiti (see Section 2.3.2) and QWERTY and optimised Soft Keyboards (see Section 2.3.2). Thus, a classification or survey of current mobile text entry methods is useful both as an introduction to the field and also as a source of inspiration for future work. Figure 2.1 shows the structure of the classification presented in this report and was based on a similar diagram from Isokoski (1999).

The most basic point of differentiation between mobile text entry methods is in the input device they use. Most current mobile text entry methods can be described as either stylus-based or key-based (MacKenzie & Soukoreff 2002). Methods based on these input devices are the focus of the classification presented in this report. Many other possible input devices exist, including speech recognisers and eye-trackers, but as these devices are primarily used in niche applications rather than for general-purpose text entry, they are not included in this classification. MacKenzie & Soukoreff (2002) and Isokoski (1999) present comprehensive surveys of mobile text entry methods and discuss methods based on a wider range

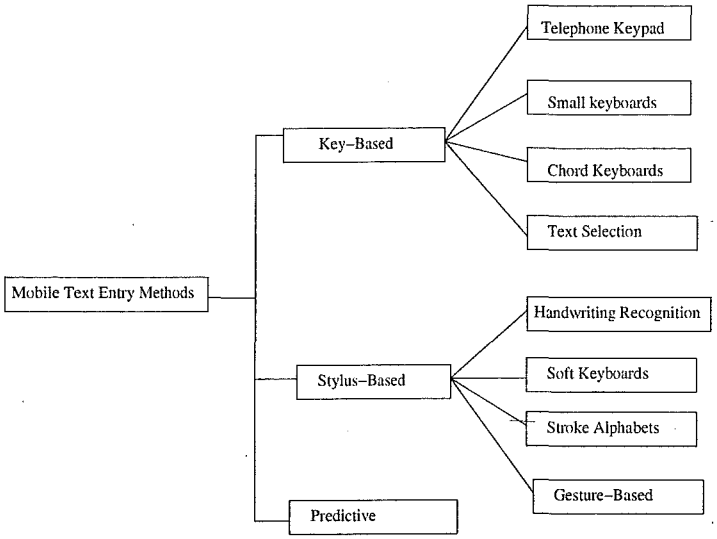


Figure 2.1: The structure of the classification.

of input devices.

Aside from quantitative measures like words per minute or error rates, there are many other ways to characterise mobile text entry methods. Other important characteristics include whether an input method is operated with one or both hands and the degree of eye-focus it requires. MacKenzie & Soukoreff (2002) introduced the term *focus of attention* (FOA) to characterise the attention demands placed on a user by a text entry method. For example, for an expert touch typist, performing a text copy task (see Section 2.1.1) on a desktop QWERTY keyboard is single FOA task, because the typist need only attend to the source text; their high level of expertise with the input method removes the need for them to look at either the keyboard or display.

Many different characterisations of what properties a mobile text entry methods should possess have been made. The developers of T-Cube (see Section 2.3.2) say a method that wishes to allow fast input should offer the user an easy way to get started and room for learning seamlessly as they write, citing the QWERTY keyboard as a classic example (Venolià & Neiberg 1994). MacKenzie & Soukoreff (2002) suggest a reasonable objective for any mobile text entry method is to produce machine-readable characters at a speed acceptable to users.

2.3.1 Key-Based Methods

Touch-typing speeds on a desktop QWERTY keyboard range from 20 wpm to over 60 wpm for expert typists (MacKenzie & Soukoreff 2002, Shneiderman 1998). The necessary reduction in the size or number of keys available on mobile devices (due to space and weight considerations), means touch-typing is usually not practical nor possible. This does not prevent users from having high expectations about the text entry speeds that should be achievable with key-based mobile text entry methods, either immediately or with minimal practice (MacKenzie & Soukoreff 2002).

Another significant expectation held by users is that they will be able to enter a large set of characters (MacKenzie & Soukoreff 2002, Isokoski 1999). Meeting this expectation, while still maintaining a reasonable balance between the number of keys used and the size of each key, is a major challenge for the developers of key-based methods. (Ward et al. 2000).

Telephone Keypad

The incredible growth of text messaging (see Section 1) has created a demand for efficient and easy to use mobile phone text entry methods. The standard input device for mobile phones is the 12-key keypad,

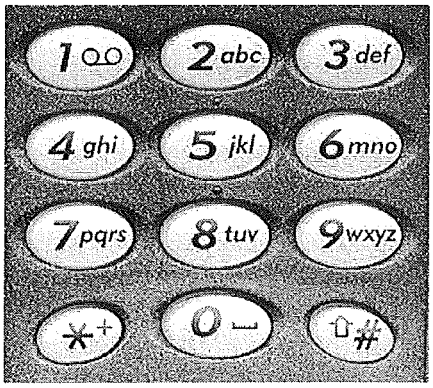


Figure 2.2: The standard twelve key telephone keypad.

shown in Figure 2.2. The keypad consists of nine number keys, 0-9, and two additional keys (* and #). Extra keys are normally provided to perform non-text entry functions, such as navigating menus. The layout of characters usually follows an ISO standard (Butts 2001). Placement of the space character varies, but it is commonly placed on the one or zero keys. Letters are arranged alphabetically, from top to bottom, left to right, using keys 2-9. To fit all 26 letters on only twelve keys, three or four letters must be assigned to each key. Thus, text input using the standard keypad is ambiguous. How the input is disambiguated is a major point of differentiation among the text entry methods that use the telephone keypad.

The original method for text entry on mobile phones is the multi-press method, also known as the multi-tap or triple-tap method. This method requires the user to press a key one or more times to enter a character. For example pressing the four key once enters a 'g', pressing it twice enters an 'h' and pressing it three times enters a 'i'. Segmentation problems occur when two characters which are mapped to the same key, such as 'h' and 'i', are entered consecutively. The system requires a way of determining where to segment the series of keypresses. The most common solutions to this problem are:

- Timeout - With this approach, a timeout is used to determine when a user has finished cycling through the characters on a key. The timeout period must elapse before another character can be entered from the same key. The duration of the timeout period is usually one to two seconds. If the users tries to enter another character from the same key before the timeout period has elapsed, the current character is overwritten. For example, to enter the word "no", the user would first press the six key twice to enter an 'n'. They would then have to wait for the timeout to expire before entering the 'o', by pressing the six key three times. If the user presses the six key after entering the 'n', but before the timeout has elapsed, the 'n' would be overwritten with an 'o'.

- 'Next' Button - Rather than waiting for a timeout to expire, a special button can be used to confirm the entry of a character and to move on to the next character. Using the example of entering "no", the user would again press the six key twice to enter an 'n'. They would then press the 'next' button to confirm the entry of the 'n'. The 'o' could then be entered with three presses of the six key. Often this method is combined with the timeout method, allowing users to choose their preferred segmentation strategy.

- 'Hold Down' - This approach totally changes the users' interaction with the keypad. Rather than repeatedly pressing a key to cycle through the characters that are mapped to it, the user instead holds the key down. Segmentation occurs automatically when a key is released. To enter "no" with this approach, the user holds down the six key to enter the 'n', releases the key and then holds it down again to enter the 'o'.

As it name implies, the multi-press method has a KSPC greater than one, unless the 'hold down' strategy is used, in which case its KSPC will be close to 1.0, though the KSPC value would be a poor

measure of the efficiency of this segmentation strategy. MacKenzie (2002) calculated the KSPC for the multi-press with next method as 2.0342.

Two-Key is another text entry method that uses the telephone keypad. Every character is entered with exactly two keypresses. The first keypress selects the group of characters the desired character belongs to. For example, to enter 'n' the user would first press the six key. The second keypress specifies the desired character's relative position within the group of characters. In the case of 'n', the user would press the two key to select the second character on the six key. This method does not work well for entering characters whose position on a key is not labeled on the the keypad or with keys which have numerous characters mappings, for example punctuation keys.

Disambiguating the input from the ambiguous telephone keypad using multiple keypresses is tedious and inefficient. An alternative approach is to allow all or some characters to be entered with a single keystroke and to perform disambiguation by some other means. Collectively these methods are called one-key with disambiguation.

T9 developed by Tegic Communications Inc.[1] is the most widely used text entry method from this category. It incorporates linguistic knowledge, in the form of a dynamic dictionary and word probabilities, to perform disambiguation. For the purposes of disambiguation a word is defined as any sequence of keypresses. The space key (usually the zero key) is used to delimit words and terminate disambiguation. For a given sequence of keypresses, the system retrieves a list of words from its dictionary that could be entered with that sequence. The list is ordered in descending order of word probabilities and the most probable word is presented to the user initially. If the initial prediction is incorrect, the user can use a NEXT button to scroll through the list of predicted words to try and find the correct word.

Dunlop & Crossan (2000) independently developed a predictive text entry similar to T9. They reported an analysis of their built-in dictionary which showed that 87.5% of the words in the dictionary were entered with a unique sequence of keypresses and therefore would always be predicted correctly. The reason for so many words having unique keypress sequences is that for any given keypress sequence only the meaningful letters combinations are included in the dictionary. The T9 dictionary is proprietary, so similar analysis can not be performed, but it is not unreasonable to assume that similar results would be found. MacKenzie (2002) performed a similar analysis, using a standard corpus rather than the contents of the T9 dictionary, when calculating a a keystrokes per character value for T9, which was reported as 1.0072. Their analysis confirmed that the NEXT key was required only infrequently. The reported KSPC figure is predicated on the "generous" assumption that only words in the built-in dictionary will be entered, which is probably not a realistic assumption Grinter & Eldridge (2001).

Predictions are made after each non-space keystroke, so for a word that is n keystrokes long, n separate predictions are made. The reason for this is that each individual keystroke is ambiguous, so predictions made for a keystroke can change based on subsequent keystrokes. This display instability is very confusing for novice users as they can not connect what appears on the display with what they (thought) they entered (Butts 2001). It also makes it difficult to correct errors as it is not obvious at what point the error occurred.

If the word being entered is not in the dictionary then disambiguation will fail. In this case the user has the option of entering the word into the dictionary, making it available for future predictions, or not. In either case, the word must be entered using the multi-press method.

T9 is not the only commercially available dictionary-based disambiguation system. Other examples include eziText and eziTap, developed by Zi Corp[2] and iTap developed by a subsidiary of Motorola[3]. eziText and iTap offer word completion as well as word prediction. No formal evaluations of these methods has been conducted.

LetterWise, developed by Eatoni Ergonomic[4], also uses linguistic knowledge to disambiguate keypresses made with the standard telephone keypad. Instead of using a stored dictionary of words and their associated probabilities, LetterWise stores a database of letter-prefix probabilities, derived from an analysis of a standard corpus (MacKenzie, Kober, Smith, Jones & Skepner 2001). For example, if the current letter-prefix is "th" and the user presses the three key, the most likely next letter in the given context is 'e'.

[1] `http://www.t9.com/`

[2] `www.zicorp.com`

[3] `http://www.motorola.com/lexicus/html/itap.html/`

[4] `http://www.eatoni.com`

Like T9, a list of predictions, ordered by probability, is generated after each non-space keystroke. The predictions are made on a letter-by-letter basis rather than for an entire word. This means there is no display instability, as new keystrokes do not change the result of previous keystrokes. If the predicted letter is not the correct one, then users can scroll the list of predicted letter with a NEXT key. The list will be three or four characters long depending on the key whose entry is being disambiguated.

LetterWise does not differentiate between dictionary and non-dictionary words. Any word can be entered, though some words may require additional presses of the NEXT button. For example, the word "thd" would be entered with the keystroke sequence ¡8, 4, 3, NEXT¿. MacKenzie, Kober, Smith, Jones & Skepner (2001) calculated that 50.1% of words, from a set of about 65000 words drawn from the British National Corpus, could be entered without having to press the NEXT key. They also calculataed the KSPC for LetterWise as 1.150.

WordWise is a linguistically optimised predictive text entry method, also developed by Eatoni Ergonomics (MacKenzie, Kober, Gutowitz, Jones & Skepner 2001). By chording an auxiliary key, similar to a shift key on a standard keyboard, with the keys used to enter letters, some letters on the standard telephone keyboard can be entered unambiguously. Only one letter per key was selected for unambiguous input to make the layout easy to learn and use. The eight letters chosen for unambiguous entry were 'c', 'e', 'h', 'l', 'n', 's', 't' and 'y'. This particular combination was chosen to minimise the system's lookup error rate and query rates (MacKenzie, Kober, Gutowitz, Jones & Skepner 2001). The query rate is how often the same keystroke sequence yields multiple words, while the lookup error rate is how often the desired word is not first in the list of alternatives in a query.

For example, the two key is mapped to the letters 'a', 'b' and 'c'. If the two key is pressed while holding down the auxiliary key, then a 'c' is entered. If it is pressed without holding down the two key, than either an 'a' or 'b' is entered, depending on the preceding context. Entire words can be entered unambiguously and each unambiguous keypress in a sequence of keypresses reduces the number of potential matches from the dictionary, increasing the likelihood that the first prediction made will be correct. For example, when entering the word "no", the 'n' is unambiguous so only the words "nm" and "no" need to be considered. WordWise still faces the same problem as T9 when entering non-dictionary words.

Small Keyboards

Many mobile devices include a small physical keyboard to provide text entry facilities. The keyboards usually use the QWERTY layout to allow for skill transfer from desktop computing, though the reduced size of the keyboards do not allow touch typing. Usually two-fingers or thumbs are used to type on the keyboards.

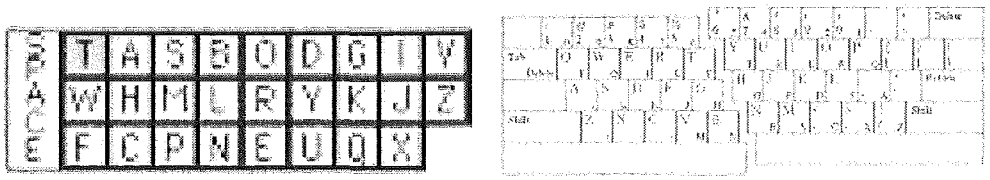
Matias et al. (1996) proposed a QWERTY-like keyboard that was reduced in size but still used touch-typing skills. The keyboard is known as the Half-Qwerty keyboard[5] and has been developed as a commercial product. As it name suggests, the Half-Qwerty keyboard is a standard QWERTY keyboard split in half vertically. Either half can be used for text entry and the characters on the other half are accessed by chording keys with the space bar. For example, if the right-half of the Half-Qwerty keyboard is being used a 'j' would be entered by pressing the 'f' and 'space' keys simultaneously. The Half-Qwerty keyboard has the advantage of only requiring one handed entry. Matias et al. (1996) suggest that the non-dominant hand should be used, leaving the dominant hand free for other tasks. Eyes-free operation is possible, as with standard keyboards. The Half-Qwerty keyboard is shown in Figure 2.3(b).

Chord Keyboards

Chords keyboard reduce the number of keys that need to be included in a keyboard by using different combinations of keys (chords), pressed simultaneously, to enter characters (Gopher & Raij 1988). For example, an eight key chord keyboard provides 256 unique chords, enough to enter any ASCII character (Isokoski 1999).

Gopher & Raij (1988) performed a comparative evaluation of a traditional keyboard with one- and two-handed chord keyboards. After thirty-five hours of practice the entry rates for all three keyboards were still improving. Gopher & Raij predicted that the chorded keyboards would have a lower ceiling on the

[5] `http://www.halfqwerty.com/`



(a) (b)

Figure 2.3: The (a) *FOCL* (MacKenzie & Zhang 1997) and (b) *Half-Qwerty* (Matias et al. 1996) key-based text entry methods

highest text entry rate that could eventually be achieved, because they provide less opportunity for parallel preparation of strokes than a traditional keyboard.

Compared to QWERTY keyboards chord keyboards are not widely used for general purpose text entry. They are mainly used in situations where text entry speed is critical, such as stenography. Isokoski (1999) attributes this to the QWERTY-keyboards appearing deceivingly easy to use, because they have a separate key for each character, but make it hard to become a skilled typist.

Text Selection

Some mobile devices, such as pagers, have so few keys that the traditional approach of key-based text entry methods, to map characters to keys, becomes impractical. An alternative is to use the available keys to select characters from an on-screen character set. While eye-focus is required while navigating through text selection methods, one-handed operation is possible (MacKenzie & Soukoreff 2002).

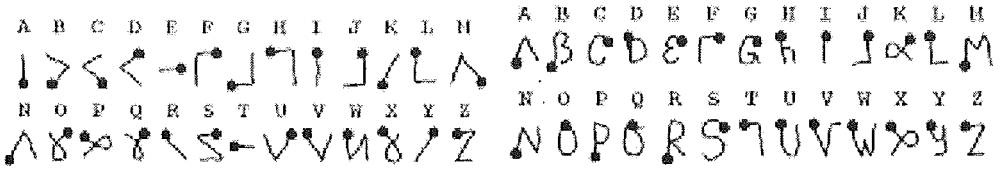
Five-Key text entry methods use UP, DOWN, LEFT and RIGHT arrows to naviagate an onscreen character set and a SELECT key to ouput the currently selected character. The character set is arranged in rows, like a traditional keyboard (MacKenzie & Soukoreff 2002). The numbers of rows and columns used in the layout varies, but usually the number of columns is larger than the number of rows (MacKenzie & Soukoreff 2002).

Bellman & MacKenzie (1998) developed an optimised Five-Key method called Fluctuating Optimal Character Layout (FOCL). Only lowercase letters and the space character were included in the layout. The FOCL method uses knowledge of the last character's input position to rearrange the character layout to place more likely characters (based on usage frequencies of letter-pairs) closer to the current cursor position (MacKenzie & Soukoreff 2002). This rearrangement is designed to minimise the keypresses required to select the next character. A *snap-to-home* cursor was used, which meant the cursor returned to a predefined position after the entry of each character. The home position was the top left corner to provide for natural searching of the layout from left to right and also because it is the natural starting position for writing in English. Three-rows were used to display the characters on the display screen. A three column-high space bar was placed in a fixed position at the left of the layout. The basic structure of the FOCL layout is shown in Figure 2.3(a)

An initial, comparative evaluation of Five-Key methods using the FOCL layout and a fixed, QWERTY layout showed no significant difference in the average text entry speeds or error rates after about ten hours of practice (Bellman & MacKenzie 1998). Participant's achieved an average entry rate of 10 wpm with the FOCL layout and were still improving at the end of evaluation. Bellman & MacKenzie were of the opinion that the participants needed more practice with FOCL before the reduction in the number of keystrokes made would outweigh the increase in visual scan time.

2.3.2 Stylus-Based Methods

Stylus-based methods combine a direct pointing device, the stylus, and a touchscreen or digitising tablet to facilitate text entry. Some methods substitute a user's finger for the stylus (MacKenzie & Soukoreff 2002).



(a) (b)

Figure 2.4: Portions of the (a) *Unistrokes* and (b) *Graffiti* alphabets. The black dot indicates the starting position of the stroke (MacKenzie & Zhang 1997).

The entry task is usually unimanual, with the other hand idle or used to support the device (MacKenzie & Soukoreff 2002).

Stylus-based interfaces are virtual in nature, which allows them to be displayed on demand, unlike the physical interfaces used with key-based methods. This allows multiple stylus-based methods to be made available on a single device. For example, Palm[6] personal digital assistants (PDAs) include both the Graffiti unistroke alphabet and a Soft Keyboard. This is not usually possible with devices that use key-based methods.

Handwriting Recognition

The combination of handwriting recognition and a digital device would seem to provide a natural extension of handwriting on paper. Unfortunately, the technology for online recognition of natural handwriting (cursive text) has lagged behind user expectations. Humans are able to deal with handwriting anomalies by using semantics, syntax and context to aid recognition. Handwriting recognition technology has slowly improved it ability to do the same by exploiting context, dictionaries, constrained symbol sets, user profiles and training (MacKenzie et al. 1994).

Handwriting recognisers take as their input a trail of digital ink, which they must segment into distinct characters for processing by the recogniser (MacKenzie & Soukoreff 2002). For handwriting recognition, a stroke is defined as a single movement that can be extracted from the digital ink trail (Isokoski 2001). The more constrained the user input is the easier the segmentation and recognition process becomes. Segmentation problems occur when trying to differentiate symbols composed of similar strokes. The need to solve difficult recognition and segmentation problems means processing natural handwriting requires complex recognisers (MacKenzie & Soukoreff 2002).

The performance of recognition technology has improved to a point where it might be acceptable for expert users, but its lack of immediate usability might discourage novice users from investing the time and effort to reach that level of expertise (MacKenzie & Soukoreff 2002).

While the text entry interface is embedded in the display with handwriting recognition systems, eyes-free operation is usually not possible because of errors in recognition; recognition rates of 87% to 93% for expert users have been reported indicating that a relatively large number of errors occur even for experienced users (MacKenzie & Soukoreff 2002). Even with perfect recognition, the performance of handwriting recognition systems would be limited by human factors and lag behind that of alternate text entry methods (MacKenzie & Soukoreff 2002).

Stroke Alphabets

Single-stroke alphabets are designed to overcome some of the problems that occur when using the Roman alphabet for text entry (Isokoski 1999). Rather than trying to mimic traditional handwriting completely, single-stroke alphabets aim to maximise the advantages of stylus-based text entry on a digitised display.

[6] `http://www.palm.com/`

These alphabets are also known as *unistrokes*, which refers to the fact each character is enter using a single stroke. In the context of stylus-based text entry, a stroke is a defined as a motion that begins when the stylus touches the surface of the digitised display and ends when it is raised (MacKenzie & Zhang 1997). The term unistrokes is not used here as it is easily confused with Unistrokes, the original single-stroke alphabet developed by Goldberg & Richardson (1993).

The strokes in a single-stroke alphabet are designed to be as well-separated as possible. This allows them to be robustly segmented even when written sloppily (Goldberg & Richardson 1993). The approach to ensuring a stroke alphabet is well-separated is ad-hoc (Goldberg & Richardson 1993). The degree of separation that exists between the strokes in a single-stroke alphabet is initially determined through visual inspection. When an acceptable level of separation has been reached, the stroke set is empirically evaluated. Recognition errors from the evaluation provide feedback about which strokes need to be separated further.

Another design criterion for single-stroke alphabets is that they should be fast to write. This is easily achieved by mapping the most frequently used characters to simple strokes, which are faster to draw than curved or bent strokes.(Goldberg & Richardson 1993)

MacKenzie & Zhang (1997) identify two disadvantages of single-stroke alphabets. Firstly, a set of new strokes must be learned and, secondly, the number of strokes or symbols must be kept reasonably small.

Overcoming the first problem requires a trade-off between stroke simplicity and easy learnability (Isokoski 1999). The more the strokes in a stroke alphabet resemble regular handwriting the easier they are to learn. If the strokes are too similar to traditional handwriting, then much of the advantage of using stroke alphabets is lost. Input efficiency for stroke alphabets is maximised by using the simplest strokes that can be processed by the recognition algorithm (Isokoski 1999).

The second problem can be overcome by exploiting the affordances provided by the input environment to reduce the size of the alphabet, as it is easier to maintain separation between strokes with a smaller set of strokes. Modes can be used to enter characters not explicitly mapped to strokes in the alphabet, such using a shift key to allow uppercase and lowercase letters to be entered with the same symbol (Goldberg & Richardson 1993). With digitised display strokes along the same axis in different directions, such as top to bottom and bottom to top can be differentiated, which is not the case with traditional handwriting (Goldberg & Richardson 1993).

Most stroke alphabets can be described as character-level alphabets because they are designed to allow one character per stroke. Word-level alphabets, designed to allow the entry of complete words with one stroke, have also been developed (Isokoski 1999).

Several character-level stroke alphabets have been developed and many of them now enjoy widespread use. With these alphabets, the symbols are spatially independent, so they may entered one on top of another. Text input and text display are separated, so only the text display needs to be legible for humans (Isokoski & Raisamo 2000). This has the advantage of reducing the writing space that is required on the display (enough space for one character is all that is required) and also reduces the amount of wrist movement required when compared to traditional handwriting (Goldberg & Richardson 1993). The most important benefit is that eyes-free entry is possible (MacKenzie & Soukoreff 2002). An important property of character-level stroke alphabets is the similarity of the strokes in the alphabet to the letters they represent, which is a measure of the learnability of the alphabet.

MacKenzie & Zhang (1997) reported a heuristic for calculating a stroke alphabet's "inherent accuracy", which is the extent to which the strokes match the letters they represent. The degree of similarity is found by visual inspection. The strokes can match either lowercase or uppercase letters. The final measure is calculated by weighting each match by the probability of the corresponding letter appearing in English text and then summing the weighted matches.

Unistrokes were the first stroke alphabet developed (Goldberg & Richardson 1993). As the name implies, each symbol in the alphabet can be written with a single stroke. The alphabet is based on five basic symbols, each of which has four possible orientations and can be written in two different directions. This provides an alphabet size of forty strokes, which is enough to enter all lowercase letters but not enough to provide a stroke for all the characters users would want to enter. Goldberg & Richardson (1993) suggested using modes to enter additional characters, such as uppercase letters. The space character, the most frequently used character in text input, is symbolised by a dot, which is entered with a tap of the stylus. The Unistrokes used to enter lowercase letters are shown in Figure 2.4(a).

Despite being quicker to write and easier to recognise than traditional handwriting, Unistrokes have

not been commercially successful. The most probable reason for this is that the strokes in the alphabet overemphasize recognition and writing efficiency at the expense of learnability (MacKenzie & Soukoreff 2002). Even a cursory glance at the Unistroke alphabet in Figure 2.4(a) shows that few of the Unistroke symbols could be considered as mnemonics for the letters they represent. A calculation of Unistrokes' "inherent accuracy", according to the procedure used by MacKenzie & Zhang (1997), illustrates its lack of similarity with the Roman alphabet. The eleven matches identified ('c', 'i', 'j', 'l', 'n', 's', 'o', 'u', 'v', 'x' and 'z') yielded a weighted accuracy of only 35.12%. This result values indicates that learning the Unistrokes alphabet will be time consuming, which reduces its immediate usability.

Graffiti is a single-stroke alphabet developed by Palm Inc., for use in their family of PDAs (MacKenzie & Zhang 1997). The Graffiti stroke alphabet includes strokes for entering punctuation, numbers and symbols in addition to letters depicted in Figure 2.4(b). Case switching and backspace are supported by mode switches, which are made using special strokes

The most immediately obvious difference between Graffiti and Unistrokes is the the similarity of the Graffiti symbols to the characters they are meant to represent (see Figure 2.4(b)). MacKenzie & Zhang (1997) calculated Graffiti's weighted, inherent accuracy as 79.2%, which is over double that of Unistrokes. Users still need to learn the symbol set, but its high degree of similarity to the Roman alphabet should make this quicker and easier (MacKenzie & Zhang 1997). The symbols in the alphabet are not as well-separated as Unistrokes, which increases the complexity of the recogniser, potentially making them slower and more inaccurate (Isokoski 1999). MacKenzie & Zhang (1997) finding that participants were able to enter Graffiti symbols with 97% accuracy after only five minutes of practice, confirms the intial impression that Graffiti trades stroke simplicity for easy learnability. The success of Graffiti as a commercial product would seem to suggest that users place a greater value on learnability than efficiency (Isokoski 1999).

Jot is a stroke alphabet developed by Communication Intelligence Corporation[7] and licensed by Microsoft for use in mobile devices (MacKenzie & Soukoreff 2002). Jot is very similar to Graffiti and recognises Graffiti strokes, in addition to its own strokes, which include single stroke and multiple stroke symbols. Jot has never been formally evaluated.

The Minimal Device-Independent Text Input Method (MDTIM) allows characters to be written individually with a single simple stroke or in groups using more complex single strokes (Isokoski & Raisamo 2000). The strokes are built from gestures in the four primary compass directions: up, down, left and right (MacKenzie & Soukoreff 2002). These gestures were chosen because they can be entered with almost any input device. This design choice reflects the fact that the goal of MDTIM is not to optimise performance with a certain input device, unlike the majority of mobile text entry methods. Instead, it is intended as a universal text entry method. The developers of MDTIM envisage it being used as an alternative text entry method for a very large range of devices, rather then being used as the primary input methods on any one device.

A calculation of MDTIMS's "inherent accuracy", according to the procedure used by MacKenzie & Zhang (1997), shows it has a low level of similarity with the Roman alphabet. The nine matches identified ('b', 'd', 'l', 'n', 'o', 'q', 'r', 't' and 'u') yielded a weighted accuracy of 41.9%. This lack of similarity is one affect of having only four basic gesture directions to create strokes with which made it difficult to create strokes similar to some characters. Unlike Unistrokes, which also suffer from a lack of similarity, MDTIM do not provide a corresponding gain in efficiency. Isokoski & Raisamo (2000) argue MDTIM's low learnability and efficiency will be mitigated by the advantages of device-independence.

Gesture-Based Text Entry

MacKenzie & Soukoreff (2002) define gestural-based input text entry methods as frameworks within which informal stylus motions (gestures) are interpreted as characters, words or operations. The stylus is well suited for use with gesture-based inputs methods as it is able to produce the precise movements that are required (Kabbash et al. 1993). Text entry with these methods is a dragging task, which Kabbash et al. (1993) found to be error-prone when performed with the stylus because users often inadvertently lift the stylus tip off the digitised surface while dragging. This would effect T-Cube and Cirrin, where lifting the stylus is used to confirm a selection and enter a space, respectively, but not Quikwriting.

[7] www.cic.com/products/jot/

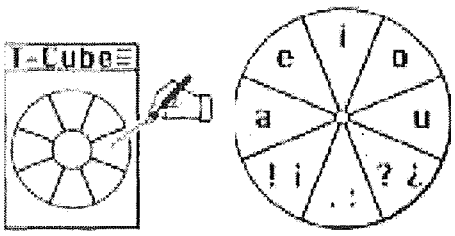


Figure 2.5: (a) The T-Cube Starting Target and (b) One of the T-Cube Gesture Mappings (Venolia & Neiberg 1994).

Word-level gestural text entry methods allow the entry of complete words with only a single stroke. This reduces the number of inter-stroke gaps, which is a major constraint on the achievable text entry speed with character-level gestures and stroke alphabets (Mankoff & Abowd 1998).

Cirrin is a word-level, single stroke input method (Mankoff & Abowd 1998). It attempts to achieve a good balance between speed and accuracy by subtituting user precision for ease of recognition by the system. The Cirrin interface is shown in Figure 2.6(b).

To spell out a word, the user traces out a path that crosses the circumference of the circle at the appropriate points, starting from the middle of the circle. Placing the characters along the circumference of the circle allows words to be spelt out with a single stroke, without entering unwanted keys. The use of a circle minimises the average distance between any pair of letters. Mankoff & Abowd (1998) also suggested columns as one possible alternative layout.

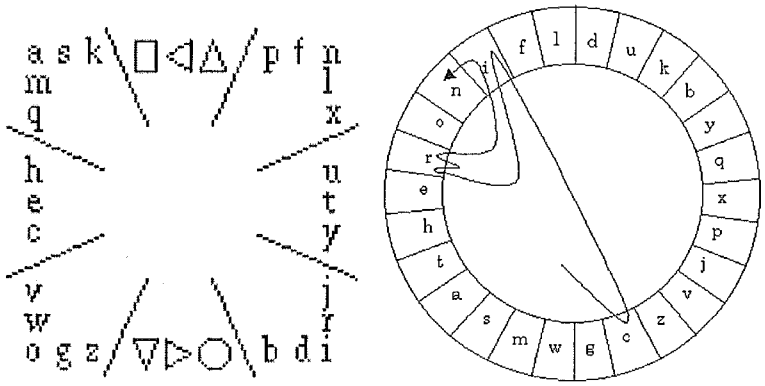
The interface supports the entry of a given alphabet, such as the Roman alphabet, directly. The space character is entered with either pen up or pen down events. Other common characters and operations must be performed via an auxiliary character-level input technique. Cirrin can not be operated in an eyes-free manner, as even expert users must track the path of the stylus through the interface.

With Quikwriting, text is entered with continuous gestures (Perlin 1998). The term continuous gestures refers to the fact that the stylus need never be lifted from the display surface or stop moving. Entire words or even multi-word texts can be written with a single continuous gesture. Other stylus based methods, like Cirrin or T-Cube, use discrete gestures, with the stylus being lifted between characters or words.

The writing area in Quikwriting is split into a three by three grid, creating nine zones. The zones are numbered one through nine, from left to right, top to bottom, starting with the top-left zone. The central zone, zone five, is called the resting zone. Each character gesture starts and finishes in the resting zone. This character-level segmentation is what allows the use of continuous gestures. The size of the Quikwriting writing area makes it impossible to display the entire alphabet at once, so it is split into four character sets: lowercase, capitals, punctuation and numeric. A single character set is displayed at any given time, the default being lowercase. Eyes-free operation of Quikwriting is not possible if users correct errors as they go, though expert users may be able to rely on only tactile feedback (MacKenzie & Soukoreff 2002, Isokoski 1999).

Each character has a major and minor zone. Its major zone is the zone in which it appears in one of the four character sets. Its minor zone refers to its relative position within its major zone. To enter a character, the stylus is dragged from the resting zone to the major zone that contains the required character. If the character's minor zone is not the same as its major zone then the stylus must be dragged to its minor zone, without entering the resting zone. The stroke is ended (for segmentation by the recogniser) when the stylus is dragged back to the resting zone. To speed text entry the most frequent gestures have the same major and minor zones.

To illustrate how Quikwriting works, consider entering an 'f', from the lowercase character set (the top-left character set in Figure 2.6(a)). Its major zone is zone three. Its relative position within zone three is top-centre, which indicates its minor zone is zone two. Thus to enter the character, the stylus must be



(a) (b)

Figure 2.6: (a) Quikwriting (Perlin 1998) and (b) *Cirrin* (MacKenzie & Soukoreff 2002).

dragged from the resting zone to zone three, then to zone two and finally back to the resting zone.

Character-level gesture text entry methods are similar to stroke alphabets (see Section 2.3.2) in that one character is entered with each stroke or gesture. With the gesture-based methods, however, users need to learn a set of gestures that are interpreted as characters from a source alphabet rather than a set of strokes that represent those characters.

T-Cube is an interface for entering text on pen-based computers using an alphabet of flick gestures (Venolia & Neiberg 1994). The alphabet is self-disclosing as it is built into the interface. The method uses a combination of nine pie menus (which are placed in the working area of the screen rather than in a menu-bar) and flicks in eight possible directions (horizontal, vertical or diagonal) to yield 72 different gestures, each of which is mapped to a character or operation. A study of the efficiency and accuracy of pie menus with different combination of menu depth and breadth found the combination of a menu depth of two and breadth of eight, which is used in T-Cube, would be efficient but more error-prone than combinations with a lower menu breadth (Kurtenbach & Buxton 1994). Figure 2.5(b) shows the layout (gesture mappings) used.

Figure 2.5(a) shows the starting point of the interface, which is always visible. The user depresses the stylus in one of the nine possible regions within the target to indicate which pie menu they wish to select from. The corresponding pie menu is only displayed if the user keeps the stylus depressed within the target region for 0.3 seconds or if the interface is in training mode. Selection is made with a flick in the direction of the item they wish to select, regardless of whether the pie menu is displayed or not. Entry is confirmed by raising the stylus. When they are displayed, the pie menus are offset from the center of the target, so they are not obscured by the users' hand.

The layout of characters and operation across the nine pie menus required a trade-off between predictability and efficiency. The developers of T-Cube placed a greater value on predictability when designing the layout, but made efficiency improvements where possible. They do not claim that the final layout is optimal.

While the interface makes heavy use of audio feedback in addition to visual feedback, eyes-free operation is not possible because it is difficult to select from the nine initial target slices without looking at the screen (Isokoski 1999).

Soft Keyboards

A soft or virtual keyboard is an image of a keyboard displayed on a touch-screen. Users can enter text by directly tapping on the keyboard with a stylus or finger. Text entry on a soft keyboard is a one-handed, eyes-focused activity (MacKenzie & Zhang 2001). The advantages of soft keyboards include their simplicity and that they are displayed on demand rather than continuously (MacKenzie & Soukoreff 2002).

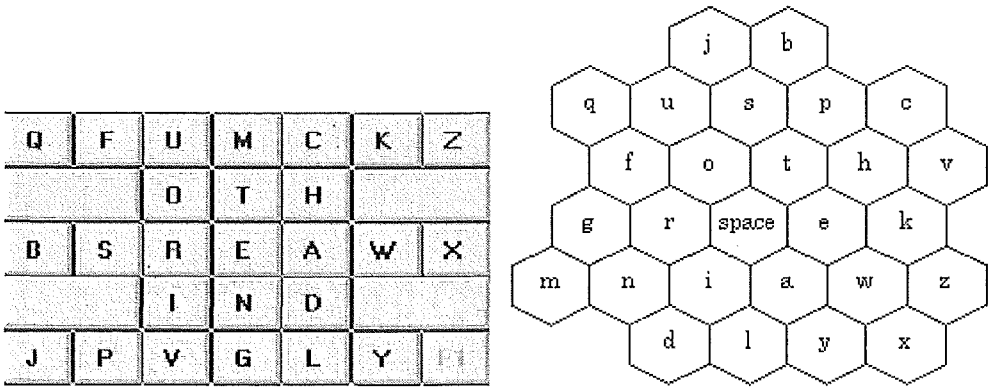


Figure 2.7: The (a) *OPTI I* (MacKenzie & Zhang 1999) and (b) *Metropolis* soft keyboard layouts. (Zhai et al. 2000).

Unlike physical keyboards, no standard layout has been established for soft keyboards. Comparative studies have found that participants with no experience using soft keyboards but who do have experience using desktop computers, can achieve substantially higher initial rates of text entry with a QWERTY layout than with alternative layouts (MacKenzie et al. 1999). In situations where soft keyboards are used for infrequent text entry by causal users, such as with touch-screen based information kiosks or vending machines, than using the QWERTY layout can provide immediate usability, by utilising the skill transfer from desktop computing (Isokoski 1999).

The skill transfer to soft keyboards for users familiar with the QWERTY layout in a desktop environment, suggests that visual scan time is the dominant component in determining text entry rates for novice users of soft keyboards (MacKenzie & Zhang 2001). For both expert and novice users, it would be expected that the time to tap a key would be minimal. Simple experiments have suggested an average time of 127ms (MacKenzie & Zhang 1999). Therefore, the key components of the task are movement time and visual scan time.

Soft keyboards are limited to serial input because touchscreens can only accept a single point of input at a time (Isokoski 1999). According to Isokoski (1999) this lack of parallelism means even expert users of soft keyboards can never reach text entry speeds of the same magnitude of expert users of bimanual, key-based methods. These two factors suggest that the expert performance advantage of layouts that minimise input device movement may be large enough to persuade users to invest the time and effort required to reach that level of expertise. The optimal layouts still needs to provide adequate immediate usability.

Unlike commercial layouts, most proposed soft keyboards layout consider only the entry of the lowercase letters and spaces. Full functional keyboards would require a way of performing functions and entering additional characters, either through the provision of additional keys or some other means.

The lexical ordering of the Roman alphabet would be expected to be as familar as the QWERTY layout to novice users (though not for text entry) thus reducing their visual scan time (MacKenzie et al. 1999). The immediate usability of the alphabetical layout has been lower than expected, despite the supposed familiarity of the ordering. This has been attributed to alphabetical keyboards being laid out in multiple rows (Zhai & Smith 2001). Unlike the QWERTY layout where the starting and ending letters of each row are an integral component of the ordering, users' familiarity with alphabetical ordering is as a continuous ordered sequence, not as a set of discrete ordered rows. Users' must first learn the breakpoints used in the keyboard (which depends on the row size used) before the effects of the skill transfer are realised.

OPTI I is an optimized keyboard layout designed using Soukoreff & MacKenzie's movement minimastion model. Aside from minimising movement, the design of layout was guided by a the need to have no dead space between the keys, for the keyboard to be rectangular in shape to fit in typical application win-

dows and to use only square shaped keys (except for the space bar which could be rectangular) (MacKenzie & Zhang 1999).

The final layout (see Figure 2.7(a))chosen was the one which provided the highest predicted expert performance, after substantial trial and error (MacKenzie & Zhang 1999). The ten most frequently used letters were placed in the centre of the keyboard, while the ten most frequently used letter-pairs were assigned the top ten keys. Four identical space bars were evenly distributed within the layout. The remaining letters were assigned to the remaining keys. The optimal space bar to use depends on the character preceding and following the space.

The Metropolis keyboard were generated by using the algorithm of the same name. The algorithm is a Monte Carlo method used to search for the minimum energy states in statistical physics (Zhai et al. 2000). Using a computerised approach developing an optimised layout, allowed the developers to consider a wider range of candidate layouts than would be possible with human trial and error. The algorithm arranged the keys (atoms) so that the total energy (movement time) of the molecule (the keyboard) was at the minimum (Zhai & Smith 2001). The devevlopers of the Metropolis keybaord have also devevloped other optimised layouts using a similar approach (Zhai et al. 2000, Zhai & Smith 2001).

Predictive Input

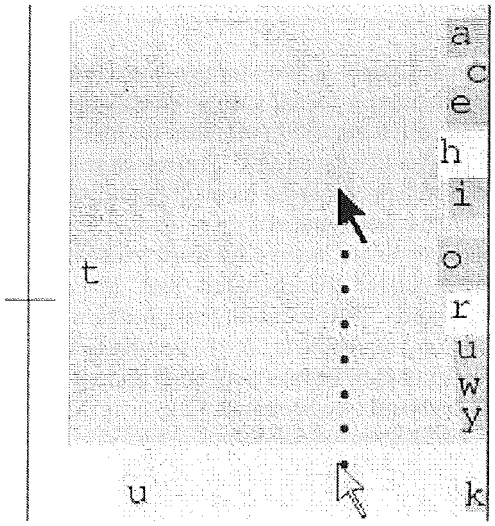


Figure 2.8: The Dasher interface after entry of a *'t'* (Ward et al. 2000)

Predictive input technologies use language models to try and predict what text the user is trying to enter. To use a simple example. if a user enters the prefix "th", a word completion system using a model of English would predict the next character as 'e', which is the most common suffix for the given prefix. Such systems make better use of the redundancy in languages, by not requiring the entry of all or most characters in words (Ward et al. 2000). Darragh et al. (1990) identify three categories of predictive system: letter anticipators which make frequent letter-pairs easier and faster to enter, word completers which amplify inputs by offering words or phrases based on an initial prefix of one or more letters entered by the user and hybrid systems which combine the functionality of the systems from the first two categories.

Language models are used in the design of letter anticipators by identifying the most frequently used letter-pairs. The layout of the interface should make the entry of these letter-pairs as efficient as possible, while still maintaining learnability. With word completers, the language model is used to predict word suffixes in real-time.

The language models used should be adaptive, meaning they learn from the text that is entered. In the case of static, dictionary-based methods, word frequencies counts should be updated, while systems with dynamic dictionaries should also allow for words to be added and deleted from the dictionary based

on usage frequencies. The technique used to generate predictions or completions must be fast enough to support interactive usage, while also being efficient in terms of resource usage.

Dasher is a predictive text interface that incorporates language modeling and two-dimensional continuous-gestures to speed up text entry (Ward et al. 2000). The twenty-six lower case letters from the Roman alphabet and the space character ('_') are displayed initially, in an alphabetical layout from top to bottom, with each character enclosed in a rectangle. Characters are entered by moving the pointer through the required character's rectangle. The display zooms in as the pointer approaches a letter, expanding its surrounding rectangle. Figure 2.8 shows the state of the interface after the entry of a 't'. Within the expanded rectangle possible extensions of the string prefix entered so far are displayed, again in alphabetical order. The heights of the rectangles surrounding each possible extension correspond to their probability in the current context for the given language. For example, in Figure 2.8, the the box around the 'h' is taller than that around the 'y', because the string 'th' is much more probable than the string 'ty' in English.

Dasher is a letter anticipator, but the developers are investigating adding word completion through use of a dictionary. Dasher can not be operated in an eyes-free manner.

The language model used by Dasher has to be efficient as it must generate numerous probabilities each time the screen is updated. The text entered is fed back into the language model to adjust future probabilities based on the user's vocabularies. To make even characters with very small probabilities relatively easy to enter, all probabilities were boosted by a fixed amount and then renormalised.

Ward et al. (2000) reported text entry speeds of up 34 wpm from an initial study. The interface is still under development.

POBox is a word completing text input method for pen-based devices (Masui 1998). Every non-space character entered by the user generates a search string, which is used to search for matching words in a built-in, static dictionary. The search algorithm searches for matches based on spelling, pronunciation or shape (for pictographic languages like Japanese). The search generates a dynamic menu of up to ten candidate words. If the correct word appears in the menu, it can be selected. Otherwise, entry can continue until a match appears or the word has been entered.

The words in the completion menu are ordered by word frequency and context (phrase or character). For example, if the first character entered after the word "user" is 'i', then the word "interface" will be at the top of the menu of predicted words. The probability assigned to words in the dictionary increases if the word is selected. Approximate string searching is used to generate candidates even if no exact matches are found. The predictive technology could be coupled any other primary input method, though in the original version it was used with a soft keyboard (see Section 2.3.2).

Chapter 3

Fastap Evaluation

Despite their drawbacks, empirical evaluations remain the most effective way to assess the performance of new text entry methods, both in isolation and in comparison to existing methods. This is especially true for mobile phone text entry methods, as attempts to use predictive models to evaluate their performance have proved unsuccessful (James & Reischel 2001, Butts 2001).

A comparative, empirical evaluation was conducted to measure the performance of a new mobile phone text entry method called Fastap. The goal of the evaluation was to measure the performance of the Fastap, T9 and multi-press with timeout mobile phone text entry methods when entering four different types of text and with three different levels of user training. The development of the Fastap interface is ongoing, so the evaluation was formative rather than summative, and the findings will be used to further refine the design of the interface.

3.1 Fastap

Fastap is a new mobile phone text entry method developed by Digit Wireless Corporation[1]. While the design of the interface draws upon many aspects of current best practice in the field of mobile text entry, it differs from almost all current mobile phone text entry methods by totally discarding the standard telephone keypad (see Section 2.3.1).

In place of the standard twelve-key telephone keypad, the Fastap interface introduces the two-layer keypad shown in Figure 3.1. The top layer contains twenty-seven small buttons which are mapped to the twenty-six letters in the Roman alphabet and the space character, whose importance in the entry of English text is recognised by its assignment to a button twice as large as the letter buttons. The layout of the letter buttons is alphabetical, starting from the top-left corner of the keypad. The bottom layer of the keypad contains eighteen large keys. Numbers, punctuation symbols and functions are mapped to these keys. The top four rows of the lower layer are arranged in the same layout as the standard telephone keypad, with the bottom three layers being devoted to function and punctuation keys.

Totalled across both layers the keypad has forty-five keys, which is almost four times as many as the standard twelve-key telephone keypad. Most importantly, input is unambiguous as almost every character is assigned to a single key. The exception to this is the punctuation keys, which each have two punctuation characters mapped to them. The second punctuation character on each of the keys can be accessed by pressing the key twice in quick succession. A shift button, which operates like a caps-lock button on a standard keyboard, allows the entry of uppercase letters. In total, the keypad can enter sixty-nine characters unambiguously.

The two-layered keypad makes the bottom layers of keys much harder to access directly without accidentally hitting the surrounding keys from the top layer. To overcome this problem the keypad implements simple chording. To enter a character mapped to a lower layer key, the user simultaneously presses the four surrounding top layer keys instead of hitting the lower layer key directly. For example, to enter a 2, the user simultaneously presses the 'b', 'c', 'f' and 'g' buttons. The small size of the letter keys makes it possible

[1] `http://www.digitwireless.com/`

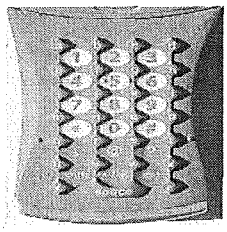


Figure 3.1: The Fastap interface.

to enter the chords with only the thumb, with no need to use complex finger combinations. The chords are easy to learn because the same basic pattern is repeated for all eighteen lowered key on the keypad; only the keys involved changes. The lowered buttons are labelled with the characters that maps to them, which removes the need for users to remember which chord maps to what character.

The keystrokes per character value for the Fastap keypad is close to one, if the entry of punctuation symbols, numbers and uppercase letters are included in the calculation and exactly one if only lowercase letters are considered. By discarding the standard telephone keypad, this KSPC value can achieved without including any linguistic knowledge, which makes the keypad much less complex than one-key with disambiguation methods (see Section 2.3.1). Any word that can be represented with the available characters can be entered, so their is no distinction between dictionary and non-dictionary words.

3.2 Method

3.2.1 Design

Two experimental designs were used. The first stage of the evaluation measures the initial reaction of participants to the three text entry methods. The design of the initial reaction evaluation was a between-subjects one-way analysis of variance (ANOVA). The between-subjects factor was 'interface' with the three levels (Fastap, multi-press with timeout and T9). A between-subjects design was used for the 'interface' factor to prevent negative skill transfer between the three interfaces and to keep the workload for each participant reasonable.

The second and third stages of the evaluation were designed to measure the performance of novice and expert users with the three interfaces when four different types of text. The experimental design used for these two stages was a mixed-factor two-way analysis of variance (ANOVA). The between-subjects factor was again 'interface', and an additional within-subjects factor 'sentence type' with four levels (Traditional, Non-Dictionary, Abbreviated, Numeric) was added to the design.

The quantitative measures of interest were the text entry speed, measured in words per minute, and accuracy, as measured by keystrokes per character. Subjective response data was recorded using the NASA TLX worksheet and five-point Likert scale questions. Participants' comments were also recorded.

Words per minute was calculated with Equation 2.2, using a value of 5.98 for the average number of characters in a word. Keystrokes per character was calculated using Equation 2.3. It was initially intended to use the minimum string distance to measure uncorrected errors. Initial analysis found the the number of uncorrected errors was too low to make the statistic worthwhile.

3.2.2 Participants

The evaluation participants were drawn from second and forth year Computer Science students at the University of Canterbury.

Thirty-four participants took part in the initial reaction and novice performance stages of the evaluation. Twenty-four of the participants also completed the training and expert performance stages. All participants in the evaluations completed a questionnaire detailing their experience with text messaging on mobile phones and with different text messaging interfaces. Four categories were provided for participants to

Interface	Beginner	Novice	Advanced	Expert	Total
Fastap	2	4	2	2	10
Multi-Press	0	3	3	2	8
T9	0	7	2	1	10

Table 3.1: Summary of evaluation participant's text messaging experience.

rate their experience with mobile text entry, based on the number of text messages they entered and sent per week using a mobile phone. The four categories were beginner (zero messages a week), novice (less than five messages a week), intermediate (five to fifteen messages a week) and expert (greater than fifteen messages a week). Six participants had never previously entered a text message using a mobile phone. Details of the participants' experience with mobile phone text entry is summarised in Figure 3.1.

3.2.3 Apparatus

Interfaces

The three mobile text entry methods that were evaluated were implemented in three different mobile phones. The Nokia 8260[2] Blue mobile phone was used to evaluate the T9 method. The Ericsson T10s[3] mobile phone was used to evaluate the multi-press with timeout method. This phone used a timeout of two seconds. Two identical Fastap prototypes were used to evaluate the Fastap keypad. The prototypes implemented only text entry functionality.

Test Sentences

Twenty-four sentences were created in each of the four sentence categories. Two initial reaction sentences and two practice sentences were also created, giving a total phrase set of one hundred test sentences. The tests sentences were designed to simulate short, realistic text messages that might be sent. All non-numeric sentences, except for the initial reaction sentences, ended with either a full stop, question mark or exclamation point. The length of the sentences varied from nine to twenty-five characters.

The traditional sentences contained only lower case letters and dictionary words. The only punctuation used was at the end of the sentences. These sentences were designed to represent basic, but still grammatically correct English text. The sentences from this category are similar to the "dictionary" sentences used in previous evaluations (Butts 2001, James & Reischel 2001).

The non-dictionary sentences added uppercase letters and non-dictionary words to the symbols and words used in the traditional sentences. The first letter in each of theses sentences was capitalised and punctuation was used in different places in the sentences, as well as at the end. The non-dictionary words were proper nouns, in particular the names of places, people and products. They were designed to seem unlikely to be included in the T9 dictionary, though some actually did. These sentences are most representative of common English text (Butts 2001, James & Reischel 2001).

The abbreviated sentences included popular abbreviations from instant messaging, SMS text messaging and email. These abbreviations have often being disparaged by researchers but Grinter & Eldridge (2001) reported that they used extensively by real users (Dunlop & Crossan 2000, Silfverberg et al. 2000). The abbreviations are by nature ad-hoc but the meaning and spelling of some abbreviations has become standardised. The abbreviations used in this evaluation were drawn from that group of standardised abbreviations. It was also intended to include so called emoticons in these sentences, but the Fastap prototype was unable to enter some of the required symbols. While these icons, such as the ubiquitous smiley face, may seem frivolous they are useful for providing contextual information in otherwise ambiguous messages.

Numeric sentences were intended to test the performance of the interfaces for entering numbers in text entry mode. The numbers used were modelled on the format of New Zealand fixed and mobile telephone numbers. All sentences in this category were nine characters long.

[2] http://www.nokia.com/
[3] http://www.sonyericsson.com/

Category	Sentence
Initial Reaction	i bought a cellphone
Initial Reaction	033667001
Traditional	i will be home later.
Non-Dictionary	I live on Ilam road.
Abbreviated	we can talk 2moro.
Numeric	039833298

Table 3.2: Examples of the test sentences used in the evaluation.

To verify that the tests sentences were representative of common English, the letter frequencies in phrase set were analysed[4]. A high correlation ($r = 0.9548$) was found between the phrase set and common English.

3.2.4 Procedure

The overall evaluation was split into two evaluation sessions. A time between the two evaluation sessions was used as a training period. The first evaluation session measured the initial reaction of the participants to the different text entry methods, after which participants were given some training and performed two practice tasks. Once the practice tasks were completed, the participants performance, as novice users, was evaluated. The second evaluation session measured expert performance.

Each session from the first part of the evaluation took about thirty minutes. The sessions were conducted over a two-week period. Before any tasks were performed the video recording of the session was setup. Video camera was used to record the screen of the interface during each task, so the text being entered could be logged. The video recording was also used as the primary timer. Backup timing was performed with a stopwatch.

After the video camera was setup, participants were asked to complete a questionnaire related to their experience with text entry on mobile phones and with different mobile phone text entry methods. While subjects were randomly assigned to one of the three interfaces prior to the evaluation, this information was important for post-evaluation analysis as participant experience was a potential confounding factor.

Before every task, participants were instructed to try and complete the task as quickly as was comfortable for them but without making too many errors. Participants were also asked to correct any errors they made during the task. The interfaces were set to text entry mode before each task. The default input mode was set to lowercase text for all three interfaces.

The basic format of each task was identical for all three parts of the evaluation. The test sentence to be entered was placed face down in front of the participants. The participants were then asked to position the interface within the recording area of the video camera. When they were ready, they were give a countdown of *3–2–1–GO*. At the end of the countdown the test sentence was turned over and the participants began entering it. Timing started at the end of the countdown and was terminated when the last character of the test sentence was entered.

The first two tasks in the evaluation session were designed to measure the immediate usability of the three text entry interfaces. The test sentences used for these tasks are shown in Figure 3.2. They were designed to be as simple as possible to enter and included only dictionary words, containing lowercase letters, and numbers. The only training given to participants prior to attempting the tasks was how to correct errors on the interface they were evaluating. A time limit of two minutes was placed on these tasks and the participants were told that if the limit was reached they would be asked to stop.

The order of the two initial reaction tasks was not rotated. This was to prevent learning effects from influencing the results for the entry of the short text sentence, which was of most interest. Thus, this task was performed first by all participants. The results for the entry of the numeric sentence were considered of lesser importance.

[4]The analysis was performed with a tool developed by Dr. I. Scott MacKenzie. The tool is available at `http://www.yorku.ca/mack/AnalysePhrases.zip`

At the completion of each block of tasks participants were asked to fill out a NASA TLX worksheet. The worksheet has six categories: mental demand, physical demand, temporal demand, effort, performance and frustration level. Each category captures a certain aspect of the demands placed on a user by a particular task. Participants were asked to record a response for each category. The responses for each category were recorded on a five-point scale from "low" to "high" (for performance the scale endpoints were labelled "good" to "poor"), with "low" being best and "high" being worst (for performance "good" was best and "poor" was worst). Participants were also asked to record any comments they had.

After each task in the novice and expert evaluations, participants were asked to respond to the statement, "it was efficient to enter this sentence using this interface", on a five-point Likert scale. The scale ranged from "disagree" to "agree", with "agree" being best. This question was designed to find the participants' subjective response to the interaction between the interface and the sentence type.

Before attempting the novice evaluation tasks, the participants were given a demonstration of how to use the interface they were evaluating. They were then asked to complete two untimed practice tasks. The practice sentences combined elements from all four sentence categories and were designed to highlight the important features of each interfaces.

The novice evaluation began after the conclusion of the practice period. It consisted of four tasks, requiring the entry of a test sentence from each of the sentence categories. The order in which the test sentences from each category were presented to participants was rotated to prevent learning effects. With four sentence categories there were 24 (4!) possible orders in which the sentences could be presented. Twenty-four sentence bundles, containing one test sentence from each category, were created and assigned one of the possible orderings. This was the order the sentences in the bundle were presented to participants for both the novice and expert evaluations. Participants were randomly assigned sentence bundles for the novice and expert evaluations, as well as for the training period.

At the conclusion of the novice evaluation tasks, participants were asked to fill out another NASA TLX worksheet and record any comments they had. They were then asked to record any overall comments they had from the first evaluation session..

The training period conducted between the two evaluation sessions was designed to provide participants with intensive practice with the interface they were evaluating. This was intended to simulate a longitudinal evaluation, which could not be performed due to time constraints. The training period was conducted over six weekdays and commenced ten days after the completion of the last initial evaluation session. During the training period every participant entered twenty-four sentences, six from each sentence category. During each training session the participants were asked to enter one sentence bundle. Participants were given access to a copy of the training guide used during the initial evaluation session as a reference. They could also ask the supervisor for help if required.

Participants attended four to six, ten minute supervised training sessions, depending on their schedules. Some participant attended two training sessions on one or two of the training days. No data was recorded during the training sessions.

The expert evaluation sessions were conducted on the three days immediately following the last training day. At the start of the evaluation session, participants were asked to record any comments they had from the training period. The expert evaluation sessions were about fifteen minutes long. Otherwise, the expert evaluation session was identical to the novice performance section of the first evaluation sessions.

3.3 Results

3.3.1 Initial Reaction

Most participant were able to complete the initial reactions tasks within the two minute time limit. The remaining participants either were unable to complete the task within the time limit or gave up before the time limit elapsed. It is unlikely that any of the participants of who could not complete the task within the time limit would have been able to complete the task even if the time limit has been larger.

For the first initial reaction task, 30 of the 34 participants completed the task within the time limit. The four participants who did not were all using the T9 interface. 27 of the 34 participants completed the second initial reaction task. Four participants did not attempt this task (two from the multi-press group and

two from the T9 group), as the task was initially intended to be performed with only the Fastap interface, but was later extended to include the other two interfaces as well. Of the three participants who attempted but did not complete the task, two came from the multi-press group, while one came from the T9 group. All Fastap participants were able to complete both initial reaction tasks within the time limit.

The mean entry speeds and error rates include only the results for participants who successfully completed the initial reaction tasks within the time limit.

The mean entry speed across all participants and interfaces was 6.56 (σ 3.902) wpm for the first initial reaction task and 5.25 (σ 3.473) wpm for the second initial reaction task.

For the first initial reaction task, the mean entry times were significantly different ($F_{2,26} = 4.503, p < 0.05$). The T9 interface was fastest (9.69 *wpm*, σ 6.699 *wpm*). The next fastest interface was Fastap (6.42 *wpm*, σ 1.705 *wpm*), while the multi-press interface was slowest (4.54 *wpm*, σ 1.307 *wpm*).

For the second initial reaction task, the mean entry times were significantly different ($F_{2,24} = 22.809, p < 0.01$). The Fastap interface was fastest (8.25 *wpm*, σ 2.255 *wpm*). The next fastest interface was T9 (3.64 *wpm*, σ 2.582 *wpm*), while the multi-press interface was slowest (1.94 *wpm*, σ 0.96 *wpm*). The mean entry speeds and standard deviations for the initial reaction tasks are summarised in Figure 3.2.

That the average text entry speeds for the second initial reaction task, for the T9 and multi-press interfaces, were lower than for the first initial reaction task is surprising as participants had the advantage of one task worth of experience with the interfaces when attempting the task.

The average text entry rate for the T9 interface, for the first initial reaction task, illustrate the potentially fast entry speeds that can be achieved with the interface, by expert users. The immediate usability of the T9 interface, however, was very poor. Participants who had no or very little previous exposure to the interface struggled to use it, while those who did have significant experience were very efficient. The high standard deviation in the average entry speed is proof of this point. The second initial reaction task proved easier for the participants using T9, with most being able to complete the task, though with varying levels of efficiency.

The multi-press interface performed poorly in both tasks, especially the second initial reaction task. Numerical entry on the multi-press interface was much more difficult than on the other two interfaces, which explains it very poor result for the second initial reaction task. For the first initial reaction task the multi-press was more usable than T9, as all participants were able to complete the task fairly efficiently.

The Fastap interface provided the best immediate usability during the initial reactions tasks, with all participants being able to complete both tasks. The results for the second task are interesting because many participants indicated afterwards they found the number buttons hard to press, having tried to press them directly rather than chording the four upper-layer buttons surrounding them. This not seem to have a significant effect on their performance.

Interface	Task	
	1	2
Fastap	6.417 (1.705)	8.249 (2.255)
Multi-press	4.537 (1.307)	1.936 (0.96)
T9	9.688 (6.699)	3.641 (2.582)

Figure 3.2: Summary of mean entry rates and standard deviations for the Initial Reaction evaluation (in wpm).

Error Rates

The Fastap interface had the lowest mean keystrokes per character value (1.086 σ 0.112) for the first initial reaction task, followed by multi-press (1.51 σ 0.783) and T9 (2.34 σ 0.466).

The Fastap interface also had the lowest mean keystrokes per character value (1.12 σ 0.259) for the second initial reaction task, followed by and T9 (2.82 σ 2.569) and multi-press (7.20 σ 4.835).

The KSPC values for Fastap interface are very close to its estimated average value of one KSPC, which indicates participants made very few errors.

The multi-press interface also performed well on the first initial reaction task, with a KSPC value lower than its estimated average value of 2.0342 KSPC (MacKenzie 2002). The KSPC value for the second initial reaction task was extremely high, reflecting the difficulties participants had entering numbers with the multi-press interface. The best way to perform this task was to hold the number keys down rather than cycling through the characters on each key to find the numbers. Many participants did not find the more efficient method until after they had made a large number of keypresses.

The KSPC for the T9 interface for the two initial reaction tasks is over double its estimated average value of 1.0072 (MacKenzie 2002). The values reflect the trial-and-error approach inexperienced users took to try and figure out how to use the interface. which resulted in many wasted keypresses being made.

NASA TLX

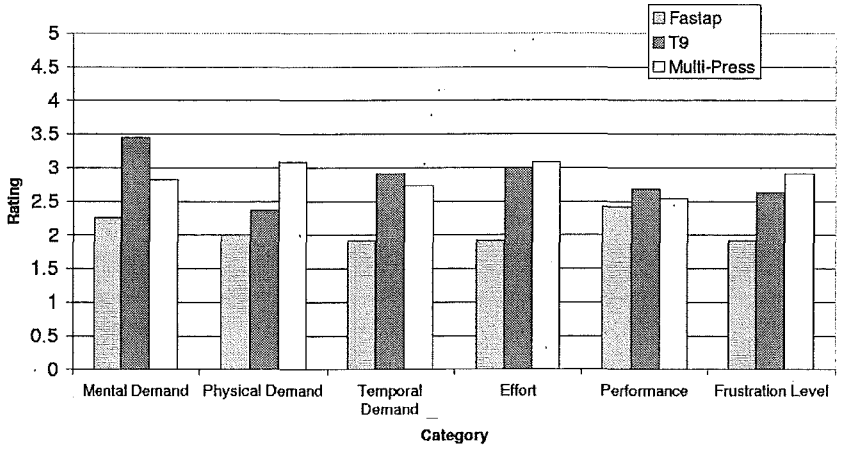


Figure 3.3: The NASA TLX ratings for the Initial Reaction tasks.

The NASA TLX rating scale was used to measure the participant's subjective response to the two initial reaction tasks. The results are summarised in Figure 3.3. Only the ratings for frustration level were significantly different across the two interfaces (Kruskal-Wallis Test corrected for ties, $H = 7.778920, df = 2, N_1 = 12, N_2 = N_3 = 11, p < 0.05$). This was surprising given the wide variation in performance shown in the quantitative results. The Fastap interface received the highest rating for all categories except mental demand. The ratings for the T9 and multi-press interfaces for the mental and physical demand categories illustrate the different approaches the two interfaces take when balancing mental and physical demand.

3.3.2 Novice Performance

The mean entry speed across all participants, interfaces and sentence types was 6.095 (σ 2.993) wpm for the novice performance evaluation.

The mean entry times for the three interfaces were significantly different ($F_{2,29} = 7.596, p < 0.01$). The Fastap interface had the highest mean entry times (7.43 *wpm*, σ 2.649 *wpm*), followed by the T9 interface (6.1904, *wpm* σ 3.8042 *wpm*) and the multi-press interface (4.56, *wpm* σ 1.613 *wpm*).

The mean entry times were also significantly different for the four sentence categories ($F_{3,87} = 13.867, p < 0.01$). The numeric sentences were entered the fastest (7.28 *wpm*, σ 3.560 *wpm*), followed by the traditional sentences (6.92 *wpm*, σ 3.366 *wpm*), the abbreviated sentences (5.54 *wpm*, σ 2.246 *wpm*) and the non-dictionary sentences (4.64 *wpm*, σ 1.738 *wpm*).

The interaction between interface and sentence category was significant ($F_{6,87} = 6.932, p < 0.01$). The mean entry speeds for the novice performance tasks are summarised in Figure 3.4.

The practice period before the novice performance evaluation was relatively brief, including only two short practice tasks, but had an impressive effect on the participants, especially those who had little or no prior experience with the interface they were evaluating. Rather than the tentative trial-and-error approach used in the initial reaction tasks, most participants were able to complete the novice performance tasks without too many problems.

The obvious exceptions were T9 participants when trying to enter the abbreviated and non-dictionary sentences. These sentences required use of some the interface's more complex features and many participants commented they had troubling remembering how to switch to different modes or to scroll through the list of predicted words. T9 performed best with the traditional sentences, which is unsurprising as those sentences are exactly the sort of text T9 is designed to enter.

The multi-press interface also improved its performance but still lagged behind the other interfaces. The average speeds were relatively consistent across all the sentence types, including the numeric sentences which had caused problems during the initial reaction tasks. The results suggest the performance of the multi-press interface is fairly insensitive to the type of text being entered.

The Fastap interface was fastest for all but the traditional sentence category. Its results were consistently good across all four sentence categories.

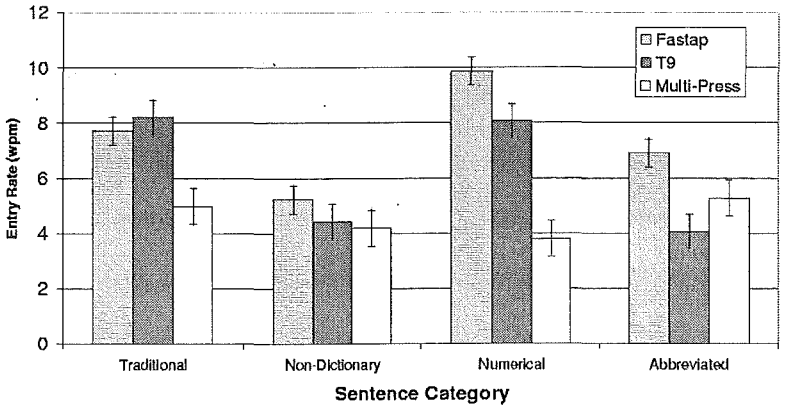


Figure 3.4: The mean entry rates for the Novice Performance evaluation.

Error Rates

The Fastap interface had the lowest mean keystrokes per character value (1.09 σ 0.150) for the non-expert tasks, followed by T9 (1.79 σ 0.884) and multi-press (2.07 σ 0.652).

The Fastap interface again achieved a average KSPC close to its estimated average value. The KSPC for the multi-press interface was also extremely close to its estimated average. This indicates the interfaces had very low error rates for the non-expert tasks. The T9 interface, however, was still above its estimated average KSPC value, indicating it had a higher error rate than the other interfaces.

Efficiency

The efficiency ratings for the traditional sentence category for the three interfaces were not significantly different
(Kruskal-Wallis Test corrected for ties, $H = 5.318, df = 2, N_1 = 12, N_2 = N_3 = 11, p = 0.070$). Fastap was rated the most efficient for entering sentences from the traditional category (4.00 σ 0.739), with T9 having the next highest rating (3.91 σ 1.136). The multi-press interface was rated as being least efficient for entering sentences from the traditional category (3.00 σ 1.265).

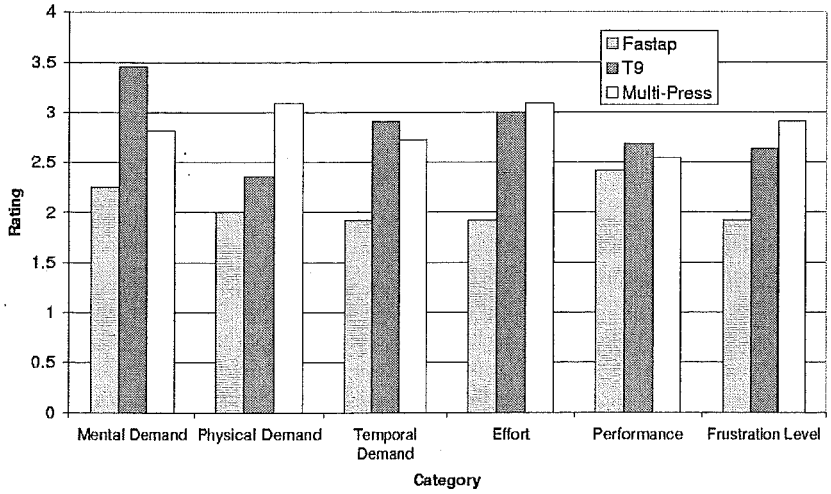


Figure 3.5: The NASA TLX ratings for the Novice Performance evaluation.

The efficiency ratings for the non-dictionary sentence category for the three interfaces were not significantly different
(Kruskal-Wallis Test corrected for ties, $H = 1.835, df = 2, N_1 = 12, N_2 = N_3 = 11, p = 0.399$). Fastap was rated the most efficient for entering sentences from the non-dictionary category (3.25 σ 1.055), with the T9 (2.73 σ 1.009) and multi-press interfaces (2.73 σ 1.104) receiving the same rating.

The efficiency ratings for the numerical sentence category for the three interfaces were significantly different (Kruskal-Wallis Test corrected for ties,
$H = 9.905729, df = 2, N_1 = 12, N_2 = N_3 = 11, p < 0.01$). Fastap was rated the most efficient for entering sentences from the numerical category (4.58 σ 0.669), with the T9 interface having the next highest rating (4.55 σ 0.522). The multi-press interface (2.818 σ 1.662) received the lowest rating.

The efficiency ratings for the abbreviated sentence category for the three interfaces were not significantly different
(Kruskal-Wallis Test corrected for ties, $H = 2.234167, df = 2, N_1 = 12, N_2 = N_3 = 11, p = 0.327233$). Fastap was rated the most efficient for entering sentences from the abbreviated category (3.50 σ 1.168), with the multi-press interface (3.00 σ 1.095) receiving the next highest rating. The T9 interface (2.82 σ 1.662) received the lowest rating.

The only efficiency rating to have a statistically significant difference across the three interfaces was for the numeric sentence category. The rating for the multi-press interface was significantly lower than the ratings for the T9 and Fastap interfaces. Multi-press participants did learn the optimal way to enter numbers on the interface, by holding down the number keys, during the practice period. As a result their performance in entering numeric sentences improved when compared to the second initial reaction task. There was no corresponding improvement in their perception of efficiency of the multi-press interface for entering numbers.

NASA TLX

There was a significant difference in the ratings for Mental Demand (Kruskal-Wallis Test corrected for ties, $H = 8.747927, df = 2, N_1 = 12, N_2 = N_3 = 11, p < 0.05$). Fastap was rated as having the lowest mental demand (2.25 σ 0.622), with multi-press having the next lowest mental demand rating (2.82 σ 0.874). The T9 interface was rated as having the highest mental demand (3.46 σ 1.036).

There was a significant difference in the ratings for Physical Demand (Kruskal-Wallis Test corrected for ties, $H = 6.214415, df = 2, N_1 = 12, N_2 = N_3 = 11, p < 0.05$). Fastap was rated as having the lowest physical demand (2.00 σ 0.953), with T9 have the next lowest physical demand (2.36 σ 1.206). The multi-press interface was rated as having the highest physical demand (3.09 σ 0.831).

There was a significant difference in the ratings for Temporal Demand (Kruskal-Wallis Test corrected for ties, $H = 6.373768, df = 2, N_1 = 12, N_2 = N_3 = 11, p < 0.05$). Fastap was rated as having the lowest temporal demand (1.917 σ 0.900), with multi-press having the next lowest temporal demand (2.727 σ 1.104). The T9 interface was rated as having the highest temporal demand (2.909 σ 0.701).

There was a significant difference in the ratings for Effort (Kruskal-Wallis Test corrected for ties, $H = 8.727790, df = 2, N_1 = 12, N_2 = N_3 = 11, p < 0.05$). Fastap was rated as requiring the least effort (1.92 σ 0.793), with T9 having the next lowest effort rating (3.00 σ 1.000). The multi-press interface was rated as requiring the most effort (3.09 σ 1.044).

The ratings are summarised in Figure 3.5.

The tradeoff between mental demand and physical demand made by the T9 and multi-press interfaces is apparent from the ratings in the mental and physical demand categories. T9 was seen as more mentally demanding than the other interfaces, while multi-press was seen as having far higher physical demands.

The effort category indicates the overall effort participants felt the interfaces required. The Fastap interface was seen as requiring a lot less effort than either the multi-press or T9 interfaces.

3.3.3 Expert Performance

The mean entry times for the three interfaces were significantly different ($F_{2,19} = 6.519, p < 0.01$).The T9 interface had the highest mean entry times (9.23 *wpm* σ 4.965 *wpm*), followed by the Fastap interface (8.86 *wpm* σ 2.406 *wpm*) and the multi-press interface (5.19 *wpm* σ 1.429 *wpm*). The mean entry times were also significantly different for the four sentence categories ($F_{3,57} = 8.549, p < 0.01$). The traditional sentences were entered the fastest (9.49 *wpm* σ 4.919 *wpm*), followed by the numeric sentences (9.03 *wpm* σ 3.945 *wpm*), the non-dictionary sentences (6.81 *wpm* σ 2.438 *wpm*) and the non-dictionary sentences (6.66 *wpm* σ 2.67 *wpm*).

The interaction between interface and sentence category was also significant ($F_{6,57} = 4.479, p < 0.01$) The mean entry speeds for the expert performance tasks are summarised in Figure 3.6.

Participants consistently commented that they felt that they had become more efficient in using the interfaces during the training period. The validity of these comments were shown by the improvement in the average text entry speeds for the each interface and sentence category.

The T9 interface displayed the most substantial performance improvement, being fastest for all but the abbreviated sentence category. For that category T9 was actually the worst performed interface. Entering the abbreviated sentences often required the use of some of the T9 interface's advanced features, such as switching to multi-press mode. The results for the abbreviated sentences suggest participants had still not mastered these features.

Fastap again performed consistently well across all sentence categories, but did not improve quite as much as T9. The performance of the multi-press interface also improved following the training period, especially for the traditional and abbreviated sentence categories. The gap between the performance of the multi-press interface and the other two interfaces continued to grow, suggesting the performance ceiling of multi-press was close to being reached.Like Fastap, multi-press performed consistently across the four sentence categories.

Error Rates

The Fastap interface had the lowest mean keystrokes per character value (1.2563 σ 0.9392) for the expert performance tasks, followed by T9 (1.4884 σ 0.6715.) and multi-press (2.1660 σ 0.6505).

The average KSPC for the Fastap and multi-press interfaces increased when compared to the novice performance tasks. Given that their average entry speeds also increased, participants might have traded an decrease in accuracy for an increase in speed. For the Fastap interface, participant's increased familiarity with the interface might have encouraged them to be less deliberate in the way they interacted with it. For T9, there is also a clear relationship between speed and accuracy. In this case, participants substantially reduced their average KSPC, which helped bring about an increase in their average entry speed. The KSPC for T9 is much closer to it estimated average value.

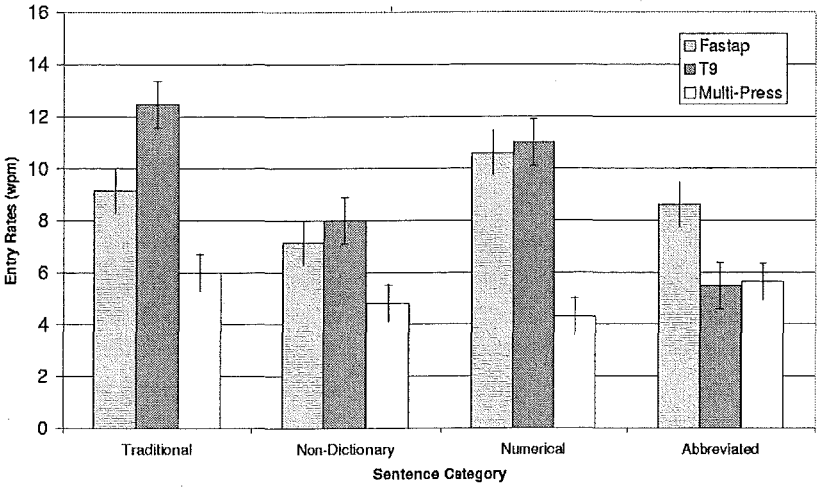


Figure 3.6: The mean entry rates for the Expert Performance evaluation.

Efficiency

The efficiency ratings for the traditional sentence category for the three interfaces were not significantly different
(Kruskal-Wallis Test corrected for ties, $H = 2.222485, df = 2, N_1 = 10, N_2 = 7, N_3 = 8, p = 0.329150$). Fastap was rated the most efficient for entering sentences from the traditional category (4.60 σ 0.516), with T9 having the next highest rating (4.50 σ 0.535). The multi-press interface was rated as being least efficient for entering sentences from the traditional sentence category (4.14 σ 0.690).

The efficiency ratings for the non-dictionary sentence category for the three interfaces were not significantly different
(Kruskal-Wallis Test corrected for ties, $H = 0.063089, df = 2, N_1 = 10, N_2 = 7, N_3 = 8, p = 0.063089$). T9 was rated the most efficient for entering sentences from the non-dictionary category (3.88 σ 0.835), with the Fastap receiving the next highest rating (3.80 σ 1.033). Multi-press received the lowest rating (2.57 σ 1.134).

The efficiency ratings for the numerical sentence category for the three interfaces were significantly different
(Kruskal-Wallis Test corrected for ties, $H = 16.775621, df = 2, N_1 = 10, N_2 = 7, N_3 = 8, p < 0.01$). The T9 interface was rated the most efficient for entering sentences from the numerical category (5.00 σ 0.000). Fastap was rated the next most efficient for entering sentences from the numerical category (4.70 σ 0.483), with the multi-press interface (2.86 σ 1.464) receiving the lowest rating.

The efficiency ratings for the abbreviated sentence category for the three interfaces were significantly different
(Kruskal-Wallis Test corrected for ties, $H = 14.762836, df = 2, N_1 = 10, N_2 = 7, N_3 = 8, p < 0.01$). Fastap was rated the most efficient for entering sentences from the abbreviated category (4.70 σ 0.483), with the T9 interface (3.25 σ 0.886) receiving the next highest rating. The multi-press interface (3.00 σ 0.816) received the lowest rating.

The perceived efficiency of entering numeric sentences was again significantly different across the three interfaces. T9 and Fastap were rated as being highly efficient while multi-press was rated as being inefficient. Numbers are relatively simple to enter on the multi-press interface, but the simplicity comes at a cost of reduced efficiency. T most efficient method of entering numbers with the T9 interface is to switch to a special number mode. Given the high efficiency ratings received by T9, it would seem participants were willing to put up with added complexity if it improved their efficiency.

The efficiency of entering abbreviated sentences also differed significantly across the three interfaces. Fastap was given a high rating for efficiency, while both T9 and multi-press received average ratings. While

the rating for multi-press was consistent with the other categories, the rating for T9 was much lower than the ratings it received for the other categories, including the non-dictionary sentences, for which it was actually rated the most efficient interface.

Expert NASA TLX

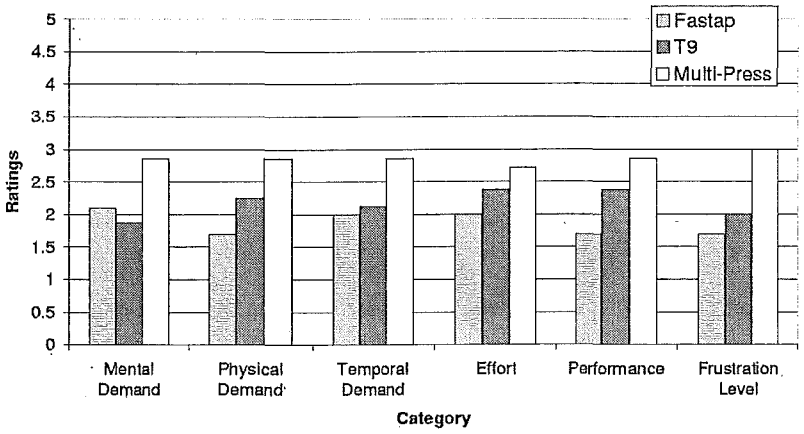


Figure 3.7: The NASA TLX ratings for the Expert Evaluations task.

The NASA TLX ratings the expert performance tasks were not significantly different. The ratings are summarised in Figure 3.7.

A clear ordering in the ratings is apparent. Fastap receives the best ratings for each category, followed closely by T9. The ratings for multi-press are lower, though not significantly so. By the end of the evaluations the participants had gained a relatively high level of expertise and familiarity with the interfaces they were evaluating, resulting in all interfaces receiving relatively good ratings in each category. This is illustrated by T9 receiving the bets rating for mental demand, which indicates that sufficient practice reduces the cognitive load that T9 places on its users.

Based on their comments, it seems that participants increased familiarity with the strengths and weaknesses of each interface had served to shape their expectation about how efficient each interface would be for a given task. While their NASA TLX ratings for any one task might be very high or very low, participants average response across all tasks were relatively similar.

3.4 Discussion

While the overall evaluation could be not be considered longitudinal, it did provide an insight into the learning curves associated with each interface. The multi-press provided quite good immediate usability, though it performed poorly when entering numbers. Its performance improved over the course of the evaluation, but not by a large amount. It would not appear that significant performance gains could be achieved with further training because of the basic inefficiencies of the method. Multi-press did perform consistently over all four sentence categories but, surprisingly, it performed best with the traditional sentences rather than the abbreviated ones. Multi-press received low ratings on the subjective response measures with participants being less than satisfied with its performance.

T9 displayed very poor immediate usability mainly because of the fluctuating nature of its display. Participants in the initial reaction evaluation struggled to find the link between the keys they pressed and the characters that were displayed on the screen. The performance of T9 did improve during the novice performance tasks but participants still had trouble with some of its more advanced features, in particular

changing between modes and scrolling through the list of predicted words. It was not until after the training period that T9's performance improved substantially, indicating substantial practice is required to master the interface. It also shows that high expert entry rates can be achieved with T9. T9 performed extremely well with the traditional and numeric sentences and even its performance with the non-dictionary sentences was relatively good. The abbreviated sentences were the most inefficient to enter because they placed an increased cognitive load on the participants.

Fastap performed well during the immediate reaction tasks and continued to improve during the evaluation. It mean entry speeds during the expert evaluation tasks were similar to T9, indicating it to has the potential to provide high entry rates for expert users. It also performed well with all four sentence types. Maybe mostly importantly, it was given the high subjective response ratings throughout the evaluation.

The traditional sentences were entered most efficiently by all three interfaces, closely followed by the numeric ones. Not surprisingly, T9 participants said the text they would actually enter with the interface would be similar to the traditional test sentences. Fastap and multi-press participants echoed the comments of the T9 participants, but also said they would use abbreviations as well. The performance of the abbreviated sentences during the evaluation suggest that the supposed efficiency gains from using abbreviations may not exist in practice. This is most likely because abbreviations include a richer set of characters than the traditional sentences and this makes them less efficient to enter. Another interesting observation was the widespread dislike of uppercase letters. This can be attributed to the poor affordances provided for entering uppercase letters in mobile phone text entry methods, when compared to those provided for entering lowercase letters.

The error rates for the Fastap and multi-press interfaces, as measured by keystrokes per second, were relatively low throughout the evaluation. The error rates for the T9 interface were high for the initial reaction and novice performance tasks, but improved during the expert performance tasks. The errors made with the Fastap interface were mainly caused by the small size of the letter buttons. Participants sometimes inadvertently chorded together several letter buttons, which resulted in a number not the desired letter being entered. With the multi-press interface errors were often caused by the participants undershooting or overshooting the desired character. Another major source of errors was using not waiting for the timeout to elapse before entering a character from the same key as the previous character. Participants who has used different segmentation techniques said they found the timeout method to be inefficient in comparison. T9 tended to be most error-prone when entering non-dictionary and abbreviated sentences which often required mode switches.

Participants switched between eyes-free and eyes-focused operation when using the interfaces. Initially, most participants used the interfaces in an eyes-focused manner, but this changed as their familiarity with the interfaces increased. During the expert performance evaluations many participants would only use look at the screen of the interface when entering a particularly tricky word or sentence, such as "cul8r" on the T9 interface.

Based on the results of this evaluation, Fastap would appear to satisfy the need for a highly efficient, easy to use mobile phone text entry method better than the two methods it was compared to.

Chapter 4

Future Work

4.1 Word Completion

Much of the current work in mobile text entry is focused on the development of text entry methods that allow, on average, any character to be entered with only a single keystroke, tap of a stylus or gesture. To reduce the the average number of keystrokes below one keystroke per character would require the provision of word completion functionality.

Word completion methods can reduce the average number of keystrokes required to enter a character below one, by not requiring that every character be entered. The actual reduction in keystrokes depends on the how many keystrokes must be entered before the correct word completion is provided and how many keystrokes are required to select the word from a list of candidates (MacKenzie 2002). The reduction in keystrokes would not automatically lead to a proportional increase in performance. The word completion systems substitute physical demand for mental demand and a poorly designed system would result in no increase or even a decrease in performance. Another disadvantage of word completion systems is that require constant visual attention to evaluate candidate completions.

Word completion would be most useful as an aid to another primary input method. Ideally an unambiguous input method, such as Fastap, would be used as this would make generating the word completions easier as each input character would contain more information. As the results of the Fastap evaluation show (see Section 3.1), users find unambiguous systems easier and less demanding to use. Adding word completion to such a system would increase the cognitive load on users, but from a much lower base.

The best practice for word completion systems if by no means settled. Important considerations, such as how many candidate words should be presented to the user and in how the primary input method and word completion system be combined, are still open questions. Word completions systems are likely to become one of the most important areas of research within the field of mobile text entry.

4.2 Contextual Enquiries

The contextual enquiry conducted by Grinter & Eldridge (2001) produced many useful findings but very few similar studies have been performed. Universities and other tertiary institutions provide a good environment for conducting field studies of mobile text entry. As students and researchers usually work at the same campus it would easier to manage the study than if participants were drawn from outside the academic world. Many empirical studies have used university students as participants so the results of a field study using the participants drawn from the same population could be compared with the findings of empirical studies to highlight any problems in the experimental methods used. Field studies would also be a good way to conduct longitudinal studies. Participants could be given the use of a new text entry in return for logging their usage and attending periodic evaluations. It is hoped that more contextual enquiries will be conducted in the future.

Chapter 5

Conclusions

This report has presented the results of an empirical evaluation that compared the two most commonly used mobile phone text entry methods, T9 and multi-press with timeout, with a new method called Fastap. The evaluation also analysed the effect that three different levels of user experience (initial reaction, novice and expert) and four different types of text (traditional, non-dictionary, abbreviated, numeric) had on the performance of the three interfaces.

The first stage of the evaluation tested the immediate usability of the three interfaces, with users being given no training before completing the two initial reaction tasks. The mean entry speeds of the three interfaces were significantly different for both tasks. T9 performed best for the entry of a short traditional sentence, while Fastap was fastest for the entry of a short numerical sentence and provided the best immediate usability. T9 displayed the most variable performance, with many of the participants using the interface being unable to complete the two tasks. Multi-press has the lowest average entry rate for both tasks. tasks.

The second stage of the evaluation measured novice performance. Participants were given a brief practice session before beginning this stage of the evaluation. The mean entry speeds were significantly different across the three interfaces and the four sentence categories. The interaction between sentence type and interface was also significant. The Fastap interface was fastest for entering sentences from the abbreviated, numeric and non-dictionary categories. T9 was fastest for entering sentences from the traditional category. Multi-press was slowest for all four sentence categories.

The third stage of the evaluation was designed to measure expert performance and was preceded by a six day training period, during which participants entered tests sentences from each of the four sentence categories, using the same interface they had evaluated during the first two stages of the evaluation. The training period was intended to quickly make the participants into expert users. The mean entry speeds were significantly different across the three interfaces and the four sentence categories. The interaction between sentence type and interface was also significant. The T9 interface was fastest for entering sentences from the traditional, numeric and non-dictionary categories. Fastap was fastest for entering sentences from the abbreviated category. Multi-press was again slowest for all four sentence categories.

Subjective responses collected during the different stages of the evaluation showed a strong preference for the Fastap interface. The subjective response to the T9 interface improved as participants gained more experience with it. The multi-press interface was rated poorly by the participants who used it.

The results of the evaluation have shown that Fastap is an intuitive, efficient and accurate method for entering text on a mobile phone that is well-liked by its users. These findings are extremely positive for the ongoing development of the Fastap interface.

Acknowledgments

I would like to thank my family and friends for all their help and support, not just this year, but whenever I needed it. Much love to the 2002 4th years for making this year a memorable experience. I would also like to thank my supervisor, Andy Cockburn, for his support, guidance and most of all his contagious

enthusiasm. The writing tutors, Jane McKenzie and Stacey Mickelbart, deserve special praise for valiantly searching for diamonds in the rough that is my prose. I would also like to acknowledge the contribution of David Levy and Digit Wireless Corporation, without whom this research would not have been possible. I wish them every success with the future development of Fastap. Finally, I would to thank all the people who took part in my evaluations for their hard work :-)

Bibliography

Bellman, T. & MacKenzie, I. S. (1998), A probabilistic character layout strategy for mobile text entry, *in* 'Proceedings of Graphics Interface '98', Canadian Information Processing Society, pp. 168–176.

Butts, L. (2001), Mobile Phone Text Entry. unpublished Honours report. *http://www.cosc.canterbury.ac.nz/research/reports/HonsReps/2001/hons_0101.pdf

Card, S. K., Moran, T. P. & Newell, A. (1980), 'The keystroke-level model for user performance time with interactive systems', *Communications of the ACM* **23**(7), 396–410.

Darragh, J. J., Witten, I. H. & James, M. L. (1990), 'The Reactive Keyboard: A Predictive Typing Aid', *IEEE Computer* pp. 41–49.

Dunlop, M. D. & Crossan, A. (2000), 'Predictive Text Entry Methods for Mobile Phones', *Personal Technologies* **4**(2), 134–143.

Fitts, P. M. (1954), 'The information capacity of the human motor system in controlling the amplitude of movement', *Journal of Experimental Psychology* pp. 281–391.

Goldberg, D. & Richardson, C. (1993), Touch-typing with a stylus, *in* 'Proceedings of the conference on Human factors in computing systems', Addison-Wesley Longman Publishing Co., Inc., pp. 80–87.

Gopher, D. & Raij, D. (1988), 'Typing with a two-hand chord keyboard: will the QWERTY become obsolete?', *IEEE Transactions on Systems, Man and Cybernetics* **18**(4), 601–609.

Grinter, R. E. & Eldridge, M. A. (2001), y do tngrs luv 2 txt msg?, *in* 'Proceedings of ECSCW'01 Conference on Computer Supported Cooperative Work Bonn, September 16–20', pp. 219–238.

Hart, S. & Staveland, L. (1988), Development of NASA-TLX (Task Load Index): Results of Empricial and Theoretical Research, *in* P. Hancock & N. Meshkati, eds, 'Human Mental Workload', Elsevier Science, pp. 139–183.

Isokoski, P. (1999), A minimal device-independent text input method, Master's thesis, University of Tampere, Tampere, Finland.

Isokoski, P. (2001), Model for unistroke writing time, *in* 'Proceedings of the SIGCHI conference on Human factors in computing systems', ACM Press, pp. 357–364.

Isokoski, P. & Raisamo, R. (2000), Device independent text input: a rationale and an example, *in* 'Proceedings of the Working Conference on Advanced Visual Interfaces', ACM Press, pp. 76–83.

James, C. L. & Reischel, K. M. (2001), Text Input for Mobile Devices: Comparing Model Prediction to Actual Performance, *in* 'Proceedings of CHI'2001 Conference on Human Factors in Computing Systems Seattle, Washington, March 31–April 6', ACM, New York, pp. 365–371.

Kabbash, P., MacKenzie, I. S. & Buxton, W. (1993), Human Performance Using Computer Input Devices in the Preferred and Non-Preferred Hands, *in* 'Proceedings of INTERCHI'93 Conference on Human Factors in Computing Systems Amsterdam, April 24–29', pp. 474–481.

Kurtenbach, G. & Buxton, W. (1994), User Learning and Performance with Marking Menus, *in* 'Proceedings of CHI'94 Conference on Human Factors in Computing Systems Boston, April 24–28', Addison-Wesley, pp. 258–264.

MacKenzie, I. S. (2002), KSPC (keystrokes per character) as a characteristic of text entry techniques, *in* 'Proceedings of the Fourth International Symposium on Human-Computer Interaction with Mobile Devices', Springer-Verlag, pp. 195–210.

MacKenzie, I. S., Kober, H., Gutowitz, H., Jones, T. & Skepner, E. (2001), Linguistically Optimised Text Entry on a Mobile Phone.

MacKenzie, I. S., Kober, H., Smith, D., Jones, T. & Skepner, E. (2001), LetterWise: prefix-based disambiguation for mobile text input, *in* 'Proceedings of the 14th annual ACM symposium on User interface software and technology', ACM Press, pp. 111–120.

MacKenzie, I. S., Nonnecke, B., McQueen, C., Riddersma, S. & Meltz, M. (1994), A comparison of three methods of character entry on pen-based computers, *in* 'Proceedings of the Human Factors and Ergonomics Society 38th Annual Meeting', Human Factors Society, pp. 330–334.

MacKenzie, I. S. & Soukoreff, R. W. (2001), Measuring errors in text entry tasks: An application of the Levenshtein string distance statistic, *in* 'Extended Abstracts of CHI 2001', ACM, pp. 319–320.

MacKenzie, I. S. & Soukoreff, R. W. (2002), 'Text entry for mobile computing: Models and methods, theory and practice', *To appear in Human-Computer Interaction* .

MacKenzie, I. S. & Zhang, S. (1997), The immediate usability of Graffiti, *in* 'Proceedings of Graphics Interface '97', Canadian Information Processing Society, pp. 129–137.

MacKenzie, I. S. & Zhang, S. X. (1999), The Design and Evaluation of a High-Performance Soft Keyboard, *in* 'Proceedings of CHI'99 Conference on Human Factors in Computing Systems Pittsburgh, May 15–20', ACM, New York, pp. 25–31.

MacKenzie, I. S. & Zhang, S. X. (2001), 'An Empirical Investigation of the Novice Experience with Soft Keyboards', *Behaviour and Information Technology* **20**, 411–418.

MacKenzie, I. S., Zhang, S. X. & Soukoreff, R. W. (1999), 'Text Entry Using Soft Keyboards', *Behaviour and Information Technology* **18**, 235–244.

Mankoff, J. & Abowd, G. D. (1998), Cirrin: a word-level unistroke keyboard for pen input, *in* 'Proceedings of the 11th annual ACM symposium on User interface software and technology', ACM Press, pp. 213–214.

Masui, T. (1998), An efficient text input method for pen-based computers, *in* 'Conference proceedings on Human factors in computing systems', ACM Press/Addison-Wesley Publishing Co., pp. 328–335.

Matias, E., MacKenzie, I. S. & Buxton, W. (1996), 'One-Handed Touch-Typing on a QWERTY Keyboard', *Human-Computer Interaction* **11**, 1–27.

Moyle, M. (2001), A Flick in the Right Direction. unpublished Honours report. *http://www.cosc.canterbury.ac.nz/research/reports/HonsReps/2001/hons_0107.pdf

Perlin, K. (1998), Quikwriting: continuous stylus-based text entry, *in* 'Proceedings of the 11th annual ACM symposium on User interface software and technology', ACM Press, pp. 215–216.

Shneiderman, B. (1998), *Designing the User Interface: strategies for effective human-computer-interaction*, 3rd edn, Addison-Wesley.

Silfverberg, M., MacKenzie, I. S. & Korhonen, P. (2000), Predicting Text Entry Speed on Mobile Phones, *in* 'Proceedings of CHI'2000 Conference on Human Factors in Computing Systems The Hague, The Netherlands, April 1–6', ACM, New York, pp. 9–16.

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

Venolia, D. & Neiberg, F. (1994), T-Cube: a fast, self-disclosing pen-based alphabet, *in* 'Conference proceedings on Human factors in computing systems : "celebrating interdependence"', ACM Press, pp. 265–270.

Ward, D. J., Blackwell, A. F. & MacKay, D. J. C. (2000), Dasher–a data entry interface using continuous gestures and language models, *in* 'Proceedings of the 13th annual ACM symposium on User interface software and technology', ACM Press, pp. 129–137.

Zhai, S., Hunter, M. & Smith, B. A. (2000), The metropolis keyboard - an exploration of quantitative techniques for virtual keyboard design, *in* 'Proceedings of the 13th annual ACM symposium on User interface software and technology', ACM Press, pp. 119–128.

Zhai, S. & Smith, B. A. (2001), Alphabetically Biased Virtual Keyboards Are Easier to Use – Layout Does Matter, *in* 'Extended Abstracts of CHI2001', ACM Press, pp. 321 – 322.