Text Entry in PDAs with WtX

Massimo Ancona* and Gianluca Quercini*

Università di Genova, Department of Computer Science (DISI), Via Dodecaneso 35, 16146 Genova, Italy

Abstract: Size is a mixed blessing for portable devices. While they feature a wide range of applications and services that can be used anywhere and anytime, their user interface is poor, hampered by the screen size. Furthermore, portable devices are meant to be used everywhere, even in uncomfortable situations (while standing up or walking for instance). Consequently, some tasks get hard to be accomplished. One of them is *text entry*, which is still far from being as comfortable as on desktop PCs. Different solutions have been envisaged, including small hardware keyboards, software (also called virtual) keyboards, predictive input techniques and handwriting recognition systems. In this paper we give a thorough overview of our text entry tool *Word Tree version X* (WtX), especially focusing on recent improvements. The new version of WtX features a handwriting recognition system and a mechanism (which we called *First-Third-Last* rule or simply *FTL rule*) forcing users to insert the characters of a word so as to improve text prediction.

Keywords: Text entry for PDAs, Text prediction, Abbreviation systems, Virtual Keyboards, Handwriting recognition systems, Comfortable Text Entry.

1. INTRODUCTION

Many years have passed since Apple Computer CEO John Sculley coined the acronym PDA (Personal Digital Assistant), referring to his new product Apple Newton. At that time, Newton was not successful for various reasons, including price. But a new era had begun. Originally, PDAs (such as the CASIO PF-3000, released in 1983) were conceived as simple electronic notebooks, with limited functionalities and computational power. Nowadays, their performances are comparable to those of old desktop PCs; processor reach frequencies up to 700 MHz, flash memories guarantee an acceptable storage capacity and operating systems provide an outstanding support to a wide range of applications. Several research projects, involving different domains such as cultural heritage and medicine, exploit PDAs to reach their goals [1, 2]. Connectivity to other devices is made possible by wireless technologies, such as Bluetooth, WiFi and cellular network. Nowadays, the boundary between PDAs and cellular phones is indeed loose; usually, devices including both the features of a cellular phone and a PDA are called *smartphones*. In a nutshell, PDAs offer so many features in so little space. As a result, people have always at hand a powerful tool to access their data and work with them. However, much work is still left to do to improve the interaction with a PDA. In particular, textual data entry in PDAs (and, more in general, in all mobile devices) is quite a challenging issue. Laptop and desktop computers users generally do not perceive this problem; whether one likes or not, a full-sized Qwerty keyboard allows fast and comfortable typing. Connecting a full hardware keyboard to a mobile device is not much of a solution; such a keyboard, in fact, is supposed to be laid down on a desk, while PDAs are supposed to be used everywhere. Small keyboards, such as those of cellular phones,are not easy to use, even just to write short text messages. Usually, PDAs are equipped with a touch screen and a stylus. Whereas the stylus allows a fast and comfortable selection of files and directories [3], the same can not be claimed for text entry. This is a critical point, as users need to input text all the time: to access a web site, to write an email, to take short or long notes and alike. A device offering almost the same functionalities as a PC must not leave aside a fast and comfortable text entry tool. In the dreams of Alan Kay, who devised the first prototype of a portable device (the Dynabook), the interaction with a PDA should be as easy as with a simple notepad; that is the so-called *pen and paper metaphor* [4]. Handwriting recognition systems (hereafter denoted as HRS) are the linchpin of this dream. Many users are still reluctant to use them, as they feel them slow and inaccurate, although good results have been reached over years of research. For this reason, an operating system for PDAs generally provide both a virtual Qwerty keyboard and an HRS.

In this paper we survey our work on WtX, a text entry tool which by now dates back a decade, and we describe in more details some recent advances. Usually, two parameters are used to evaluate a text entry tool: *speed*, which gauges the maximum number of words that can be written in a minute, and *accuracy*, which is a measure of the errors committed while writing some text. We introduce a third parameter called *comfort*. We define *comfort* as the inverse of the tiredness perceived by the user while writing a text. A text entry tool is *comfortable* when it is immediate to understand and learn, it adapts to users' needs and demands and helps the entry of long words or words difficult to spell. The latter is a particularly relevant feature for people using words from highly specialized dictionaries (doctors and technicians for example). In the last version WtX has been provided with a HRS and a mechanism which requires users to insert a word by specifying its first, third and last

*Address correspondence to these authors at the Università di Genova, Department of Computer Science (DISI), Via Dodecaneso 35, 16146 Genova, Italy; Tel: +39 010 353 6605; Fax: +39 010 353 6699;
E-mails: ancona@disi.unige.it, quercini@disi.unige.it

character before the others. This technique, denoted as *First-Third-Last* rule (or simply, FTL rule) considerably improves text prediction and does not affect comfort, as our experiments show.

2. STATE OF ART

Text entry tools for PDAs can be roughly classified into five classes, according to [5]: hardware keyboards, virtual keyboards, HRSs, gesture-based methods and voice recognition systems.

There are several types of hardware keyboards. The most common text entry method for cellular phones is the 12-key keypad, in which keys are alphabetically arranged and each key is associated to more than one character [5]. Two methods are used to entry text with such a keypad: the *multitap* method and the *predictive* method. In the first case, each key is pressed one or more times to specify the character to insert, in the second case a dictionary is used to find the words matching a given key sequence. T9 is a famous example of predictive method [6, 7]. A better solution for PDAs are portable keyboards, which are connected *via* a cable or Bluetooth. Recently, ElekTex came out with a new keyboard made of cloth, which can be rolled up and even washed. Infrared laser keyboards are also very popular. All these devices, though impressive, do not provide a satisfactory solution as they limit the portability of the PDA. Moreover, an hardware keyboard, thus a peripheral, may also be larger than the device itself.

Virtual keyboards were conceived to address portability, while providing a familiar and easy-to-use tool. Basically, a *virtual keyboard* is the image of a keyboard, whose keys are pressed by using either a stylus or fingers. The action of selecting a key on a virtual keyboard is referred to as *tap*. Usually, PDA operating systems provide a virtual Qwerty keyboard, which is displayed in the Soft Input Panel (SIP), the area of the screen (usually the bottom) devoted to text entry. Qwerty is not the only option as a virtual keyboard. It is widespread mainly because users are acquainted with it. However, the main drawback of Qwerty is that letters forming frequent digraphs in English are far from one another, which increases hand movements and decreases writing speed. Qwerty is a legacy of old typewriters, which it has been designed for. It is a common belief that Qwerty has been specifically thought to slow down writing speed. In fact, back to back pressures of two close keys in a typewriter may cause them to jam. However, Baber disagrees, claiming that the Qwerty layout is more the result of a random choice [8]. However it is, in a pen-based virtual keyboard letters forming frequent digraphs should be as close as possible, so as to increase writing speed. That is why many alternative layouts have been envisaged. Fitaly[1] is among the most known. It bases on three as much simple as clever ideas. First, unlike Qwerty, follows a portrait form factor, resulting in a more compact layout, which allows letters to be closer to one another. Second, the most used letters are located at the center of the keyboard, so as the hand movements are minimized. And, finally, there are two keys for the space character, which is the far most used one [5]. Readers interested in learning more on layouts optimized for stylus entry are referred to [9]. In this interesting survey, the authors use a model based on Fitts' law to evaluate the maximum speed achievable with the proposed keyboards. Their results show that most of them outperform Qwerty. However, the reader should keep in mind that these results refer to performances which can only be obtained by skilled users. In other words, users can do better with a keyboard other than Qwerty only after re-training themselves on the new layout. And this, of course, takes time and effort.

HRSs have received great attention from researchers, but they are still far from the results Alan Kay had dreamed of. Most commercially available systems force users to lift up the stylus after any character: this clearly makes writing unnatural, slow and stressful. Moreover, handwriting varies from person to person and is challenging to train a system to recognize all such different styles. A common solution is to provide an alphabet of symbols which the machine can easily recognize: each symbol corresponds to a different character [10]. Currently, the systems which adopt this approach, such as Unistroke, Graffiti and Jot, guarantee an acceptable accuracy, even though they force users to adapt to a new alphabet. Gesture-based systems follow this philosophy, as they associate particular gestures to characters or even words [11-13]. Usually, they are a mixed approach, which combines a virtual keyboard with gesture recognition. The virtual keyboard should help users to learn the gestures associated to each word.

Speech recognition systems would be the most effective interaction method, as they require less attention than the other techniques. In the last decade many efforts have been invested in this research field [14]; however, their accuracy can not be compared to that achievable by a virtual keyboard or even a HRS. Moreover, some situations prevent speech recognition to be used, especially when users would like to preserve their privacy (e.g. in a hospital).

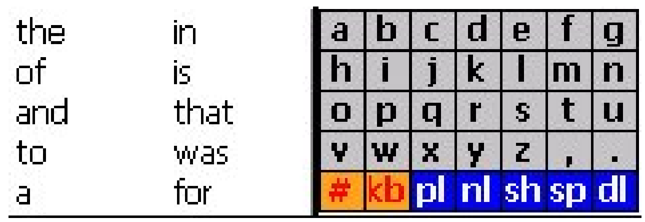
Finally, some approaches can be cited not falling in any of the previous categories. *Dasher* gets rid of both virtual keyboard and HRS and offers a zooming interface to select characters of a word [15]. In its first configuration, Dasher displays in a column all letters (and the space character) in alphabetical order. To select a character, one has to move the stylus towards it; while doing so, the interface updates in a zooming motion, showing only the letters that can occur after the first one. For instance, after selecting *t*, letters such as *b*, *c* or *g* are not displayed, as no English word begins with *tb*, *tc* or *tg*. *Tengo*[2] consists of a Qwerty keyboard having just 6 large keys; each key is associated to more than one character, as in cellular phone keypads. Having larger keys improves accuracy, as users are less likely to type the wrong key. In order to guess the word the user wants to insert, Tengo uses T9. Moreover, it suggests a set of words that are likely to occur after the newly inserted one. *Unipad* [16] and *PoBox* [17] are probably the most similar to WtX. Both rely upon word completion to improve writing speed. Unipad also support suffix completion, which allows fast insertion of several words, having just their stem in the dictionary. Unlike WtX, however, it has been tested on a Wacom PL-400 tablet, which is larger than a PDA and this allows more freedom while designing the interface.

[1]www.fitaly.com

[2]www.tengo.net

Table 1. WtX Interface with a Virtual Keyboard (a) and with HRS (b)

(**a**)



(**b**)

3. WtX MAIN FEATURES

In 1996 we started the development of WordTree (WT)[3], the father of WtX [18]; as its name suggests, WT managed the dictionary as a tree (a trie, more precisely), both from the data structure and the user interface point of view. WT was designed for supporting context and location aware applications, i.e., applications whose input data are strongly influenced by the context and the location in which such applications are executed. Typical inputs consist in few words selected from large dictionaries (≥10K elements) of predefined words, organized in a tree structure; one example are the names of treatments and diseases used in medical contexts. The Microsoft standard tree-view control was used to browse the dictionary. It is the same approach used to browse the file-system on Microsoft desktop operating systems. The nodes of the tree were labeled with letters and the leaves were those words matching the explored path. WT was released for Ms Windows CE 2.11; however, in the testing phase it came in trouble with the size of the screen. Even if browsing the dictionary by single strokes on the nodes is a fast action, every char-selection opens a new level of 26 nodes. This forces the user to scroll the window till the needed character is visible, stressing the user and decreasing the input speed.

We developed WtX as an improvement of WT: WtX extends WT by adding the capability to support also general textual input. The *x* at the end of the acronym means that the software is still far from complete. In mathematics, in fact, *x* usually refers to a variable, whose value is either unknown or bound to change. The results we present in this paper are quite satisfactory; however, there is still much room for improvement, as we will point out in the subsequent sections.

In the current release, WtX works in two modes, the *traditional* mode and the *FTL* running mode, which differs in the way users are required to write the words. In the first mode, words are written as usual by inserting each character, from the first through the last; in the second mode, a word is written by selecting its first, third and last character before the others. We will dwell upon this rule and its motivations later on in Section 4. WtX splits the SIP into a *composition area*, which contains either a small virtual keyboard (Table **1a**) or an HRS (Table **1b**) and a *selection area*, which displays up to ten words loaded from a dictionary.

The last row of the keyboard (Table **1a**) is devoted to special function keys. # and *kb* are used to switch to a keyboard containing the numbers and the punctuation marks respectively; *pl* is used to append the suffix *s* to a word to form its plural; *nl* inserts a new line and *sp* a space; finally, *dl* is used to remove the last inserted character. In the *FTL* mode, key *pl* is replaced by key *tr*, which forces WtX to quickly switch over to the traditional mode. Some strings (such as URLs and email addresses) can not be easily written by using the FTL rule; moreover, it would not make any sense to do so, as the FTL rule is meant to improve text prediction, which can not be applied to such particular strings. In Table **1b**, key *Tr* is clearly visible; key *1/2* serves to force the HRS to recognize numbers. Using WtX is straightforward: at the startup, the selection area lists the ten most frequent English words. Whenever a letter is inserted, the ten most frequent words beginning with that letter are shown in the selection area. As soon as the desired word comes up in the selection area, it can be inserted with a single tap. This also automatically inserts a space character.

3.1. Comfort

As anticipated in the introduction, WtX aims at improving *comfort*, defined as the inverse of the tiredness perceived by a user while writing a (possibly long) text. In order to make this point clearer, we briefly recall the original application of WtX. At that time, only the interface in Table **1a** was implemented. WtX was used to support doctors in a hospital to prescribe treatments to patients [19]. Words from a medical dictionary are often long and difficult to spell; drug names, for example, usually owe their names to their active ingredients or to the contraction of chemical compound names. As the dictionary is highly specialized, most of such words come up in the selection area after at most two or three taps. Furthermore, in that application WtX is context and location aware; when doctors walk from a ward to another, a dictionary, containing more specific words for the new ward, is automatically loaded, so as to improve text prediction. In this context, it is quite clear what comfort means: one inserts long and difficult words with just two or three selections and words are inserted with no errors (since they are loaded from a dictionary), keeping the user from the need of correcting them.

As claimed before, a *comfortable* text entry tool must: (a) be immediate to understand and learn; (b) adapt to users' needs and demands and (c) ease the insertion of long words or words which are difficult to spell. To match the first requirement, we designed the interface of WtX to be self-explanatory. The ten most frequent English words, present in

[3] A preliminary demo has been shown at the Eighth Biennial Conf. Int'l Graphonomics Soc., A.M. Colla, F. Masulli, and P. Morasso, eds., Genoa, Italy, Aug. 1997.

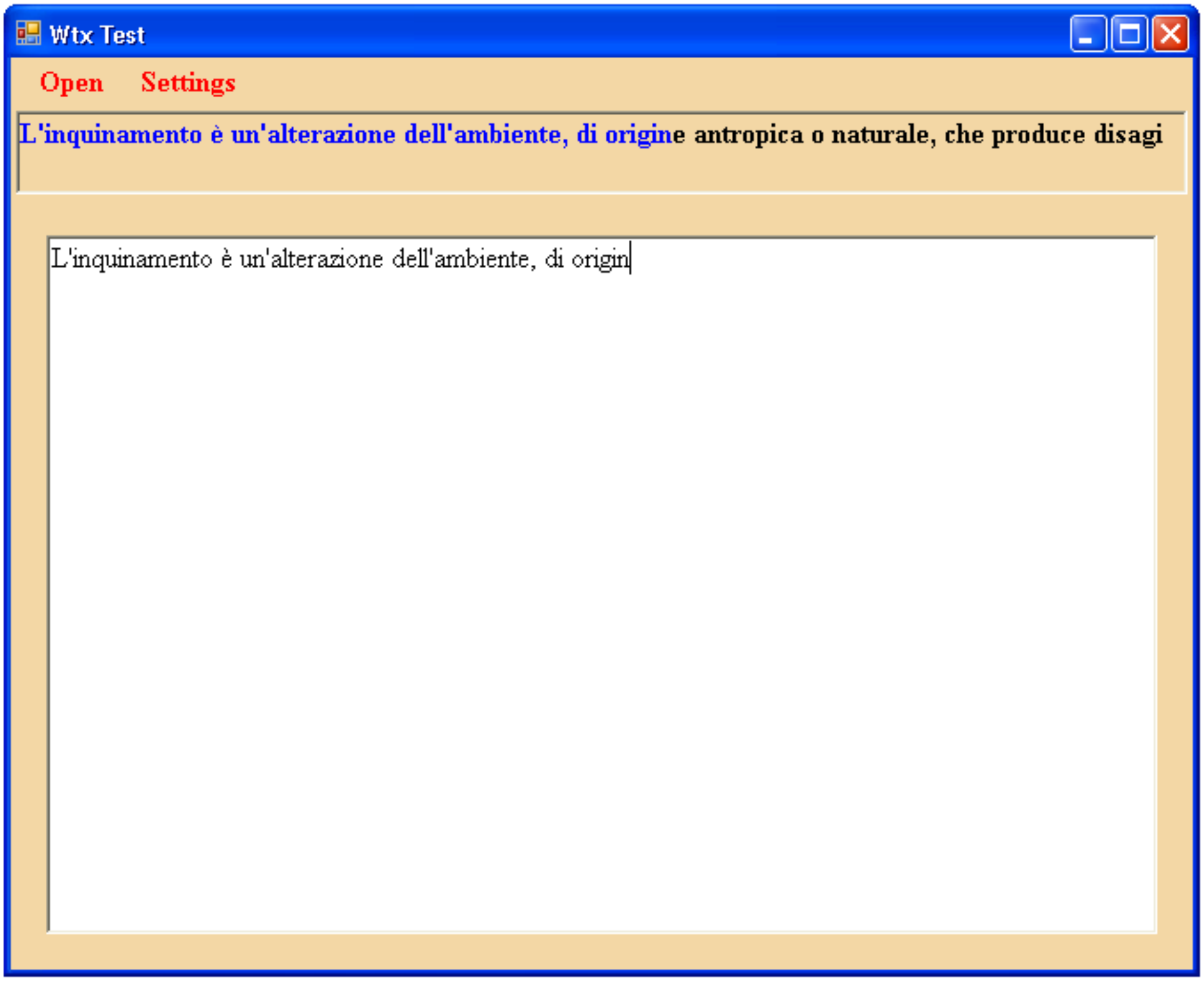


Fig. (1). Text editor used to evaluate speed variation while writing a long text.

the selection area at the startup, do not only help quick text entry, but also implicitly explain the function of the selection area. By default, the composition area hosts a keyboard rather than a HRS. The latter, in fact, is just a blank rectangular area, which communicates nothing about the role of the composition area. Rather, a keyboard is well-known to everybody as a way to insert characters in an electronic device. As for the adaptability, WtX offers a wide ranges of options, which let users personalize the interface. First, selection area and composition area can be switched, so as to help left-handed users. Next, either a virtual keyboard or an HRS can be used; this is extremely important, as there is no universal agreement on which one is the best. The choice simply depends on personal tastes of users and nothing more. Finally, the last version of WtX provides also the FTL rule, whose advantages and drawbacks will be analyzed in Section 4. Finally, to help the insertion of long words, WtX can display up to 10 words in the selection area, which, as we will see shortly after, allows dictionary words to be inserted after just two or three taps (or strokes, if using an HRS) on average. The FTL rule even improve this support.

Evaluating *comfort* is not straightforward, as it is a subjective parameter. Baber introduced the concept of *user preference*, which is highly related to what we mean by comfort [8]. User preference is mainly evaluated on the basis of a questionnaire users are requested to answer after a test. Interestingly, Baber points out that usually user preference leans towards fast text entry tools rather than accurate ones. Apparently, errors do not bother users very much. This makes sense, as even in desktop Qwerty keyboards, which users seem to like, backspace character is one of the most typed [5]. However, speed alone can not be used as a measure of comfort. In particular, speed is computed by dividing the length of a text by the time taken to produce it. In this way, lot of information gets lost. That two users achieved the same speed does not imply that they maintained the same speed. One, in fact, may have started to write very quickly and slowed down while approaching the end of the text; the other may have kept a more constant speed over all the test. We believe that speed variation over time is an important clue of the comfort perceived by the user. To support our claim we run a simple experiment[4], aiming at evaluating speed variation over time while using a tool which most of computer users deem as comfortable: the desktop Qwerty keyboard.

We asked 10 volunteers to write a given sample text, using the desktop Qwerty keyboard and a special text editor,

[4]http://www.gianlucaquercini.net/app/download/1746417016/Test.zip

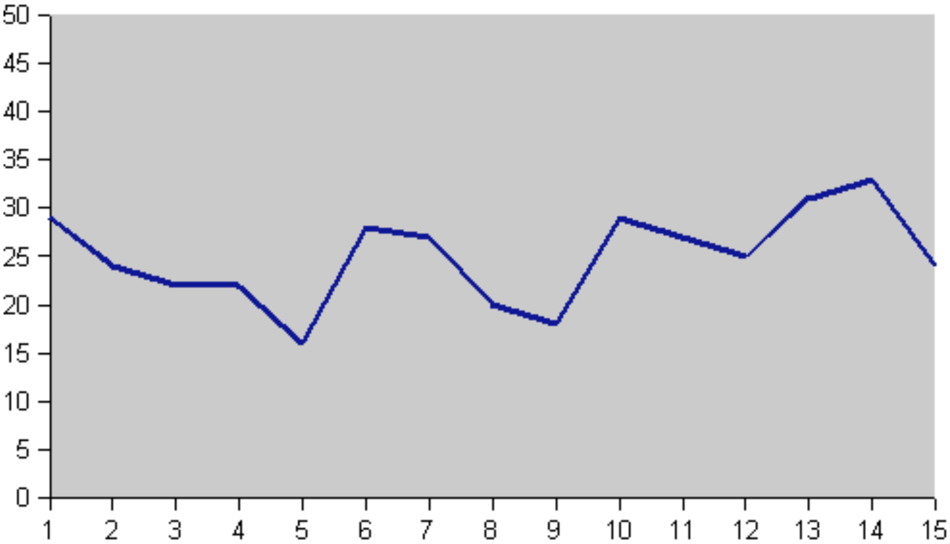


Fig. (2). Average speed variation obtained using the desktop Qwerty keyboard.

which records the number of written characters every minute. The count does not include neither space nor backspace characters. Moreover, the text does not require users to type any other special character, such as new line or tab. All participants have a good familiarity with Qwerty, as they need to use it every day. The interface of the text editor is shown in Fig. (**1**). The text strip right below the menu displays the sample text line by line; the large white text area receives users' input. The editor checks the correctness of each typed character against the sample text; if the character is that the editor is expecting, it is highlighted in blue in the text strip. Otherwise, it must be erased, using the backspace. Neither keypad arrows nor the mouse were allowed to move through the text and correct the errors. Users were required to correct their mistakes as soon as they noticed them. This sounds a severe limitation, but it mimics a real-life keyboard usage. Most people, in fact, especially well-trained typists, do not even look at the keyboard while writing; therefore, they usually remark errors as soon as they make them. Thus, the fastest way to correct them is to use the backspace, instead of moving the cursor to where the error occurred, correcting it and moving again the cursor to the proper position. As a matter of fact, when asked to grade the text editor, no participant expressed the need of using neither the mouse nor the arrows. As all participants are native Italian speakers, the sample text is in Italian, to make them feel more comfortable. Moreover, we allowed them to run the experiment on their own laptops, so they could use the keyboard they felt more acquainted with.

After the test, we asked each participant after how many minutes they felt tired; interestingly, in the majority of the cases, speed variations are more evident after that point. However, there are some results that are unclear to us. In particular, we did not find any direct correlation between the tiredness perceived by users and their average speed variation. In particular, in many cases the average speed variation is lower for slower participants, who actually declared to be more tired. This is a point yet to be addressed. What is interesting is to compare the graph in Fig. (**2**) with the one in Fig. (**8**). Both represents speed variation over time, the first obtained when using the desktop Qwerty keyboard and the second when using the virtual Qwerty keyboard on a PDA. In the second case, users claimed to be on average more tired than in the first and this results, as expected, in a higher speed variation.

3.2. The Keyboard

In the previous section we stressed the importance of comfort, as a new parameter to take into account when evaluating a text entry tool. One common complaint about virtual keyboard is related to the limited size of keys. In WtX this issue is even more evident, as the interface consists of two vertically tiled areas, which halve the space available for a virtual keyboard. Clearly, fitting a full keyboard into the composition area would result in an unusable keyboard.

Our first solution consisted in a partial keyboard, which only contained the most used characters; a special key allowed the user to switch to another keyboard, containing the remaining characters and symbols. Examples of partial keyboards are the half-Qwerty [20] and DotNote[5]. This solution, however, is unsatisfactory for two reasons. On one hand, it considerably increases the number of keystrokes, due to the continue switching between the keyboards. On the other hand, frequent changes to the interface destroy the *mental map* of the user; in other words, they confuse them.

In order to avoid switching, we decided to create a keyboard which contained at least all letters, the space character, the most used punctuation marks (period and comma) and some other control characters (shift, delete, new line). A special key (labeled as #) allows to show a keyboard containing all other symbols. The keyboard in Fig. (**3**) has been designed by allocating the most used keys close to the centre, in order to minimize the hand movements. The second keyboard (Fig. **4**) is obtained by placing letters so that to form frequent English words, such as ``the'', ``they'', ``at'', ``as'' and ``for''; moreover, space, which is the far most used character, is at the centre, as in most of the best virtual

[5]http://www.utilware.com

,	v	u	m	k	.	dl
w	h	o	n	p	z	pl
d	i	e	t	l	j	nl
y	r	a	s	b	q	sh
sp	f	c	g	x	#	kb

Fig. (3). A 34 wpm keyboard.

u	f	o	r	b	z	.
c	t	h	e	y	k	,
w	a	n	sp	l	p	dl
v	s	i	d	g	j	sh
x	q	m	#	kb	nl	pl

Fig. (4). A 39 wpm keyboard.

keyboards [9]. As we expected, the second keyboard outdoes the first one; the predicted speed, using the Fitts model is 34 wpm and 39 wpm respectively. The second one is also expected to be easier to learn, because of the presence of whole words.

Despite these good theoretical results, we used these two keyboards only in a previous release of WtX, but we discontinued them as soon as we realized that users did not appreciated them. Adapting to a new layout takes time and, as pointed out in [9], users are willing to invest their time only if the advantage they get back is worth the effort. However, we decided not to use the Qwerty keyboard, as it has a landscape form factor, which is not suitable for the WtX composition area. This is why we decided to adopt an alphabetical keyboard, which has the advantage of both being familiar to the users and compact in size.

3.3. The Handwriting Recognition System

Many people manage to write with an hardware keyboard without even looking at it. The same can not be claimed for virtual keyboards. This may dwell in the different feedback received from the keyboard. Hardware keyboards give back a direct tactile feedback, in that users feels the keys under their fingers. Software keyboards give either a visual or an auditory feedback. Tactile feedbacks are still possible (a vibration for example), but they are usually indirect, that is they do not come directly from the keyboard. Furthermore, in a virtual keyboard the boundary between two keys is not well-defined; if a user taps too close to the boundary of a key, the wrong character is likely to be inserted. Thus, a virtual keyboards demands high concentration from users, not to commit errors. Finally, there may be situations in which seeing what is on screen is just impossible (in bright sunshine for example).

Although the accuracy of an HRS is generally poor, if compared to that of a virtual keyboard, some people prefer it, as it requires less attention. In fact, letters can be written without the need of constantly watching the composition area. Therefore, while writing a letter, one can look at the selection area at the same time. This reduces the search time of the desired words and improves speed. As a result, we expect that WtX with a HRS is more comfortable than with a virtual keyboard; our test results, shown in Section 5, confirm this impression.

While using a virtual keyboard, comfort is evaluated in terms of number of taps; with an HRS, we use the concept of *stroke*, defined as the gesture done to write a letter with the stylus. ``Tapping" is definitely faster than ``stroking"; thus, a character is inserted faster by using a virtual keyboard than an HRS. However, when using an HRS, we could also decrease the size of the composition area, so that to show more words in the selection area, and this would decrease the average number of strokes.

In order to obtain a quick evaluation, we interfaced WtX to the handwriting engine provided by Windows CE. This engine is actually highly inaccurate and negatively influenced our tests. This is also due to the impossibility to force the HRS to dynamically tune the parameters used for recognizing strokes to the large contextual information maintained by WtX. In other words, each recognition system should make available to the client application (WtX in our case) mechanisms for dynamic tuning its threshold parameters used for discriminating the input keystrokes. For this reason, we did not account for the errors cause by the system. Even so, the speed was poor and clearly unacceptable.

3.4. The Dictionary

WtX was first used to select words from a dictionary of names of therapies and pathologies. Usually, these words are long and also complicated to write; in WtX they generally show up in the selection area after 2-3 taps. In the current version, the dictionary counts more than 13000 words of general English. Our first concern, when creating it, was to rigorously classify words by their usage frequencies. To this extent, we used the word frequency table based on Brown Corpus [21]. It is interesting to notice that the first 100 words in this table have a 50% cumulative frequency; this easily implies that the remaining words have a very low occurrence frequency. In general, the word occurrence frequency complies with Zipf's law, which claims that the frequency of any word is inversely proportional to its rank in the frequency table [22]. Thus, the most frequent word will

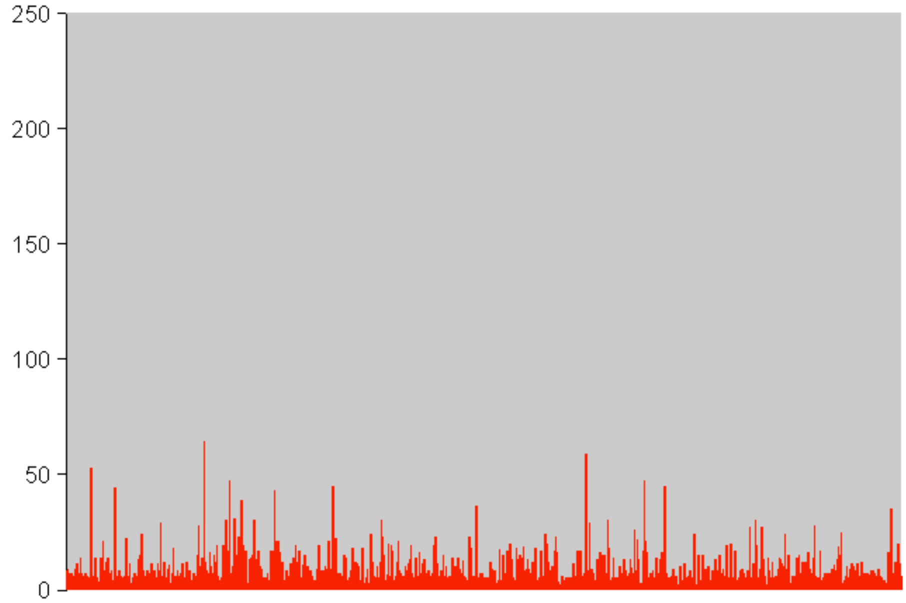


Fig. (5). Number of words matching each useful sequence in the FTL running mode.

occur approximately twice as often as the second most frequent word. Moreover, it is just enough to compute the frequency of the first 100 most used words and obtain the remaining frequencies using Zipf's law. This is just the approach we took to create our dictionary.

4. THE FTL RULE

The aim of every text prediction system is to understand the word being inserted as soon as possible. In the current dictionary, the average word length (weighted on the frequency of the words) is 4.4 characters. In the traditional running mode, the average number of taps required to insert a word present in the dictionary is 2.43. We computed this figure by using a simple Java procedure, which iterates through all the words in the dictionary and mimics the behavior of WtX. Therefore, in the traditional mode the number of taps is half of that needed if no dictionary is used.

Unless a word is very frequent, the system is unlikely to guess and suggest it to the user, after just one character has been inserted. For example, in WtX dictionary there are 800 words beginning with ``a''. It is clearly impossible to visualize all of them in the selection area; and it is not even worth it, as users would spend most of their time in looking for the desired word in the list. In an ideal setting, it would be desirable that each character sequence is matched by a as small set of words as possible.

This is the underlying idea of the FTL rule [23]. Usually, we write a word by sequentially inserting its characters, from the first one to the last one; the FTL rule forces users to write

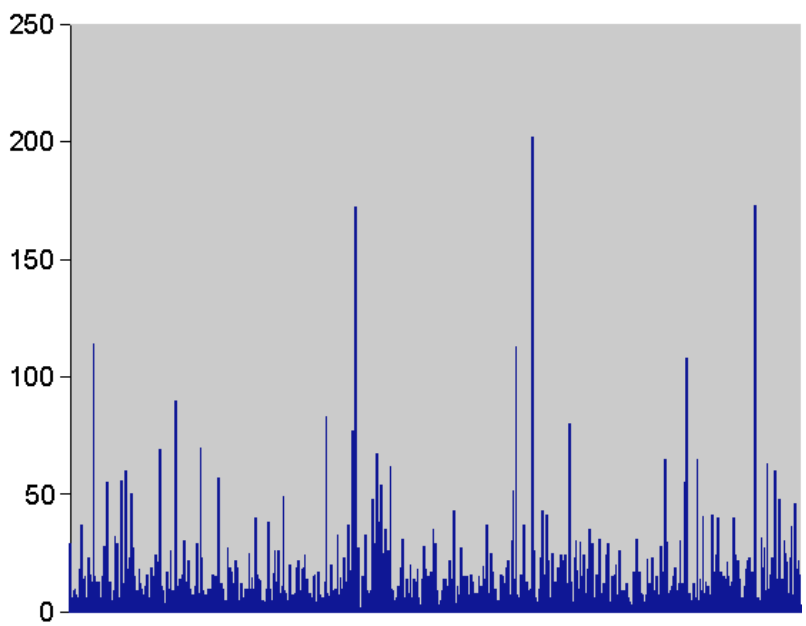


Fig. (6). Number of words matching each useful sequence in the traditional running mode.

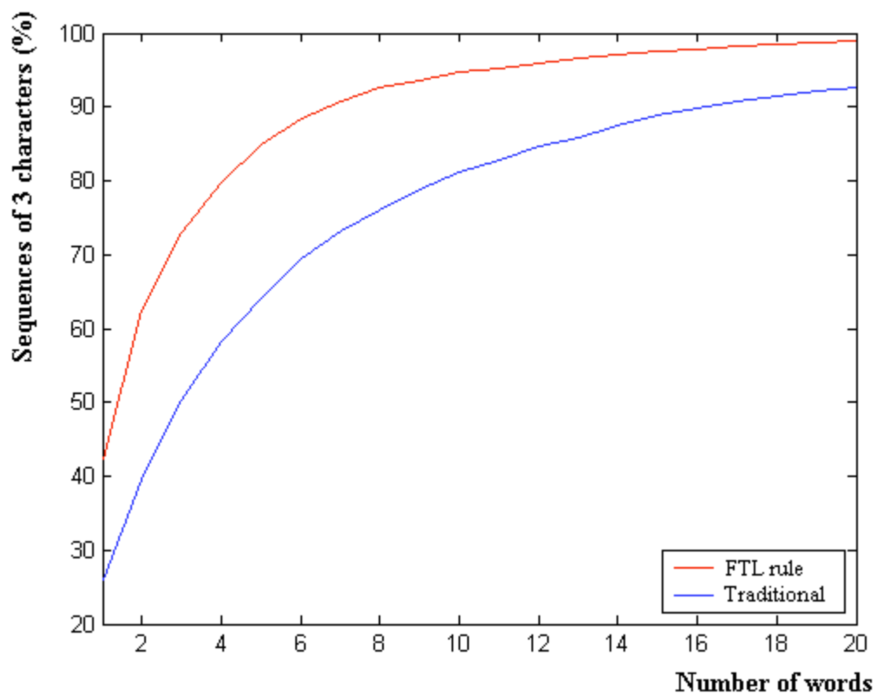


Fig. (7). Number of words matching each useful sequence in the traditional running mode.

a word inserting the first, the third and the last character. Indeed we noticed that, given a sequence of three characters, the set of words in the dictionary matching it is considerably smaller when interpreting them as the first, third and last character of a word. Figs. (**5**, **6**) explain better this point. Both graphs plot the *useful sequences* (that is sequences of three characters matched by at least one word in the dictionary) on the x-axis and the number of words matching each sequence on the y-axis. A useful sequence, interpreted with the FTL rule (Fig. **5**), is matched by 64 words in the worst case, whereas, in the traditional mode, by 202. Moreover, every useful sequence matches on average 6 words when using the traditional mode and 3 (exactly the half) when using the FTL mode. Finally, in the FTL rule the useful sequences are around 4000 and 1800 in the traditional mode, which means that in the first case the words are more equally distributed over all useful sequences.

Fig. (**7**) is even more significant, as it shows that 94% of the useful sequences are matched by less than 10 words with the FTL rule. Thus, in most cases the desired word comes up in the selection area after just three taps. This figure decreases to 81% in the traditional mode. Table **2** reports the number of taps needed to insert a word with both the FTL rule and the traditional mode: 88% of the words are inserted with less than 5 taps in the first case, only 64% in the second case. Finally, the average number of taps needed to insert a word is 2.32, slightly better than before.

The FTL rule seems to significantly improve text prediction and minimize the number of taps. Obviously, there is a price to pay for this improvement: users are required to learn a new way to write words. This is not the first time that users would be asked to adapt to a system to improve it. We saw an example right at the beginning of this paper, when talking of HRSs: Graffiti requires users to learn a new alphabet to make character recognition more accurate. T9 is another clear example. In the case of the FTL rule, the situation is a little bit more complicated, as some words can not be easily inserted, if they are not in the dictionary. In this case, in fact, after typing three characters, the user should insert the second and then the fourth, the fifth and so on. We leave to the imagination of the reader how cumbersome is inserting an email address in this way. For this reason, WtX provides a key in the composition area which let users switch between the two running modes.

Table 2. Selection Power of Both WtX Modes

# Tap	# Words Trad. Mode	# Words FTL
<2	0.07	0.07
<3	2.31	1.86
<4	14.65	29.49
<5	64.50	87.85
<6	93.12	95.12
<7	98.75	99.62
<8	99.96	99.93
<9	100.00	100.00

4.1. Abbreviation Systems and FTL Rule

The FTL rule belongs to the line of products known under the name of *abbreviation systems*. The most known one is the Fitaly Instant Text[6], which allows the insertion of words or even sentences using abbreviations. Unlike other similar systems, InstantText does not force users to

[6]http://www.fitaly.com/overview/overviewpage.htm

memorize abbreviations; this is made possible by associating to each word many abbreviations. For instance, the word ``dichlorodifluoromethane'' can be abbreviated as ``didime'', or ``dcfm'', or ``doooh'' and still other options are available. This freedom of choice, however, can be at the same time a disadvantage. When faced with a new word, in fact, users have to reflect on how to abbreviate it. Moreover, an abbreviation may match no word; this may happen either because there is no word in the dictionary matching that abbreviation or because that abbreviation is not recognized for the intended word. Users are unable to tell one condition from the other, so they may end up trying different abbreviations before realizing that no one is possible. The FTL rule limits the user freedom, but states only one method to abbreviate a word; thus no memorization is required and, if an abbreviation does not match any word, this only means that the intended word is not in the dictionary. In the last case, the user can still write the other letters of the word without the need of deleting them.

5. TEST RESULTS

Evaluating a text entry method is not straightforward. Typically, experiments are run where a group of selected participants are asked to write a text and some parameters are then evaluated. As said before, usually speed and accuracy are evaluated as objective parameters and user preference as subjective parameter. While calculating speed is trivial, there is no agreement on how to evaluate accuracy [24]. Some authors suggest to account for the number of misspelled words, other for each misspelled character. As a result, different measures have been envisaged to give accurate estimates of accuracy [5]. User preferences are usually evaluated on the basis of questionnaires, which are also an important feedback to improve the entry tool.

Due to the complexity of WtX, we conducted our tests only on beginners so far. We selected 10 participants with no skills in using PDAs and virtual keyboards and we asked them to write a 454 words English sample text using three different tools: the virtual Qwerty keyboard, WtX running in traditional mode with virtual keyboard and WtX running in FTL mode with HRS. Tests have been all run on Compaq iPAQ H3630, running a Microsoft CE 3.0 operating system.

The sample text[7] is an excerpt from one of our papers on WtX [25]. We wanted the text to be general enough to mimics a real-life usage of WtX. Therefore, the text contains punctuation marks, capital letters, acronyms and even words not in the WtX dictionary. Also, the text is long enough to challenge the tiredness of the users. 454 words mean a short text, when using a full desktop keyboard, considering that somebody write up to 70-80 words per minute; but they mean a long text, using stylus-based text entry, as beginners are not expected to write more than 15 words per minute.

As anticipated in the previous sections, we evaluated three parameters for each participant: speed, accuracy and comfort. Speed is computed as usual as number of words per minute. As for accuracy, we just accounted for the misspelled words; we also released participants from the burden of correcting the errors, which also made accuracy calculations rather simple. we had just to allow an exception with the HRS. As a standalone application, in fact, the Microsoft HRS, which WtX is interfaced to, works acceptably well, with a satisfactory accuracy. When embedded into WtX, however, it randomly occurs that it is unable to recognize characters, even if they are well written. Usually, it suffices to press the space character to make the HRS work properly again. Therefore, we asked the participants to inform us when all characters of a word were not recognized, in order not to account for such errors. Finally, comfort is computed as the average variation of speed over time (in terms of characters per minute); the larger is this value, the less is the comfort, as discussed in Section 3.1. The figures in Table **3** are averaged over all participants. Fig. (**8**) shows the average speed variation in the three cases: using Qwerty (in blue), using WtX in traditional mode (in red) and WtX in FTL mode (green).

Table 3. WtX Test Results

Session	Speed	Accuracy	Comfort
Qwerty	13.33	2.35	35
Trad. Mode + Keyboard	9.64	0.22	29
FTL + HRS	6.43	0.28	28

As expected, participants wrote more quickly when using Qwerty, as they are familiar with it. However, accuracy is low and speed variation is high, which means a low level of comfort. The low level of comfort is also due to the rudimentary text prediction system Qwerty is provided with in the Windows CE OS. Therefore, in general, each word needs to be written from the first character through the last one. With WtX, accuracy remarkably improves, as most of the words are directly selected from the dictionary and comfort benefits from it. When asked, users reported to feel less tired when using WtX with FTL rule and handwriting recognition system. This was a further evidence that speed variation is a good indicator of comfort. Anyway, we were surprised of this result, as the HRS suffers from the aforementioned bug and the FTL rule requires users to do a supplementary mental effort. In particular, participants agreed on considering the virtual keyboard too small and the main cause of their errors. Moreover, they suggested to alphabetically sort the words in the selection area, to shorten the time needed to search for them.

Finally, we further stress that these tests only concerned people who have never used WtX before. Moreover, all participants were native Italian speakers, while the sample text was in English. This is something participants complained about and we believe it had negatively affected our results. Since the feedback received from participants was positive and the results on comfort fit our expectations, we trust in that one could reach an acceptable speed (15-20 wpm) after few hours of training. We reserve to investigate this important point in our future work.

6. CONCLUSIONS AND FUTURE WORK

WtX is a text entry tool we have been developing over the last ten years. Although WtX is up and running, much work is still left to do. WtX users frequently complained

[7] http://www.gianlucaquercini.net/app/download/1747809616/sample.txt

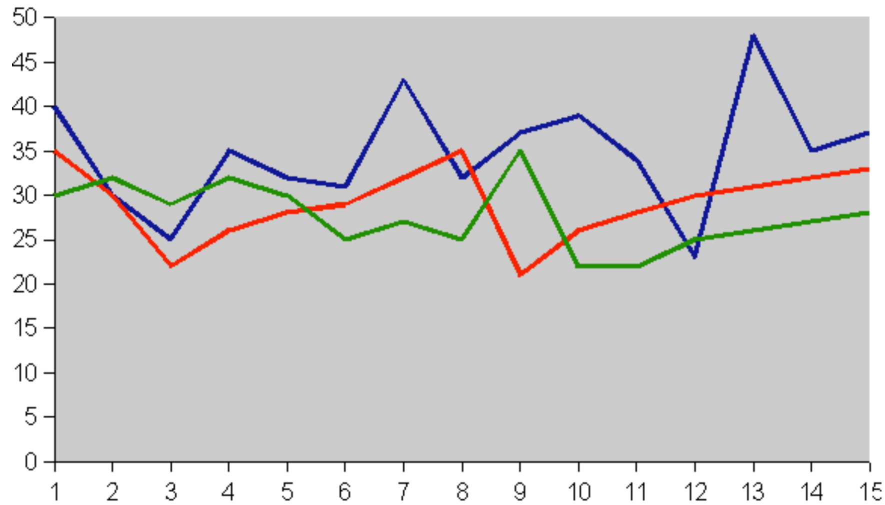


Fig. (8). Average speed variation using Qwerty (in blue), WtX in traditional mode (in red) and WtX in FTL mode (in green).

about the small size of the virtual keyboard and seemed to prefer to use the HRS. Anyway, it is not our intention to discontinue the keyboard. As repeatedly pointed out, many people stick with the virtual keyboard, as they feel HRSs slow and inaccurate. But we need to find a solution to overcome the problem of having small keys.

For now, we embedded the Windows CE HRS into WtX; this required a lot of coding, which we suspect to be the main reason of the slowness of WtX at startup. Moreover, for some unknown reason, the HRS behave differently when used as a standalone application and when embedded in WtX. We are planning to integrate a new (possibly more accurate) HRS in the next releases.

Several improvements can be applied to the dictionary, which may end up in a better text prediction. In our current dictionary, words are ranked by their frequency, but frequencies are static. That it to say that their frequency does not change based on their actual usage. This would definitely contribute to make WtX more flexible and adaptable to users' needs. It would also be desirable that the dictionary contains the stems of the words and that a mechanism helps users to append suffixes to them; this also would definitely decrease the number of needed taps. Finally, WtX should also care of multilingualism.

As for performances, the results we presented in this paper are acceptable preliminary results, but we need more accurate ones. In particular, we need to carefully evaluate the learning curve; a text entry tool can not be claimed as successful if it requires too many hours of training to outdo existing tools.

REFERENCES

[1] Virtuoso S. WardInHand - Mobile access to EPRs. In: Universal Access in Health Telematics. Heidelberg, Germany: Springer 2006; pp. 177-85.

[2] Ancona M, Dodero G, Gianuzzi V, *et al*. Exploiting wireless networks for virtual archaeology: the PAST project. In: Proceedings of the 6th International Conference on Virtual Systems and Multimedia; Arezzo, Italy, November 24-25, 2000.

[3] MacKenzie IS, Sellen A, Buxton WAS. A comparison of input devices in element pointing and dragging tasks. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. United States: New Orleans, Louisiana, April 27 - May 2, 1991.

[4] Meyer A. Pen computing: a technology overview and a vision. ACM SIGCHI Bull 1995; 27(3): 46-90.

[5] MacKenzie IS, Tanaka-Ishii K. Text Entry Systems. USA: Morgan Kaufmann 2007.

[6] Grover DL, King MT, Kuschler CA. Reduced keyboard disambiguating computer. United States patent US 5818437, 1998.

[7] Silfverberg M, MacKenzie IS, Korhonen P. Predicting text entry speed on mobile phones. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. The Hague, The Netherlands, April 1-6, 2000.

[8] Baber C. Beyond the desktop: designing and using interaction devices. Academic Press 1996.

[9] MacKenzie IS, Soukore_ RW. Text entry for mobile computing: models and methods, theory and practice. Hum Comp Interact 2002; 17(2): 14-98.

[10] Goldberg D, Richardson C. Touch-typing with a stylus. In: Proceedings of the INTERCHI '93 Conference on Human Factors in Computing Systems. Amsterdam, The Netherlands, April 24-29, 1993.

[11] Venolia D, Neiberg F. T-Cube: a fast, self-disclosing pen-based alphabet. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. Boston, Massachusetts, USA, April 24-28, 1994.

[12] Kristensson PO, Zhai S. SHARK: a large vocabulary shorthand writing system for pen-based computers. In: Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology. Santa Fe, New Mexico, USA, October 24-27, 2004.

[13] Sazawal V, Want R, Borriello G. The unigesture approach. In: Proceedings of the 4th International Symposium on Mobile Human-Computer Interaction. vol. 2411 of Lecture Notes In Computer Science. London, UK: Springer-Verlag 2002; pp. 256-70.

[14] Zaykovskiy D. Survey of the speech recognition techniques for mobile devices. In: Proceedings of the 11th International Conference on Speech and Computer. St. Petersburg, Russia, June 26-29, 2006.

[15] Ward DJ, Blackwell AF, MacKay DJC. 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. San Diego, California, USA. November 06-08, 2000.

[16] MacKenzie IS, Chen J, Oniszczak A. Unipad: single stroke text entry with language-based acceleration. In: Proceedings of the 4th Nordic Conference on Human-Computer Interaction. Oslo, Norway, October 14-18, 2006.

[17] Masui T. An efficient text input method for pen-based computers. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. Los Angeles, California, USA, April 18-23, 1998.

[18] Ancona M, Comes D. WordTree: a pen-based editor of short texts. In: Proceedings of the 9th Biennial Conference of the International Graphonomics Society. Singapore, Singapore, June 28-30, 1999.

[19] Ancona M, Locati S, Romagnoli A. Context and location aware textual data input. In: Proceedings of the 2001 ACM Symposium on Applied Computing. Las Vegas, Nevada, USA, March 11-14, 2001.

[20] Matias E, MacKenzie IS, Buxton W. Half-QWERTY: typing with one hand using your two-handed skills. In: Conference Companion on Human Factors in Computing Systems. Boston, Massachusetts, USA, April 24-28, 1994;

[21] Francis WN, Kucera H. Brown Corpus Manual. Retrieved from the website of Unifob AKSIS; 1979; Available from: http://khnt.aksis.uib.no/icame/manuals/brown/.

[22] Zipf GK. Selective Studies and the Principle of Relative Frequency in Language. Harvard University Press 1932.

[23] Quercini G. Fast text prediction in WtX: the FTL rule. Genova, Italy: University of Genoa, Dept Computer Science 2007; No. DISI-TR-07-04.

[24] Soukore RW, MacKenzie IS. Measuring errors in text entry tasks: an application of the levenshtein string distance statistic: extended abstracts on human factors in computing systems. Seattle, Washington, March 31-April 05, 2001.

[25] Ancona M, Quercini G, Locati S, Mancini M, Romagnoli A. Comfortable textual data entry for pocketPC: the WtX system. In: Proceedings of the 12th Conference of the International Graphonomics Society. Salerno, Italy, June 26-29, 2005.

Received: May 20, 2009 Revised: August 28, 2009 Accepted: September 10, 2009