A Powerful Pseudo-Syllabic Text Entry Paradigm

Francesco Curatelli \textsuperscript{a}, Chiara Martinengo \textsuperscript{b}

\textsuperscript{a} DIBE–University of Genova, Via Opera Pia 11/A, 16145 Genova, Italy
e-mail: curatelli@dibe.unige.it

\textsuperscript{b} DIMA–University of Genova, Via Dodecaneso 35, 16146 Genova, Italy
e-mail: martinen@dima.unige.it

Corresponding author: F. Curatelli
DIBE–University of Genova, Via Opera Pia 11/A, 16145 Genova, Italy
Phone/Fax: +39 010 353 2784/2777 e-mail: curatelli@dibe.unige.it

Abstract

This paper describes a powerful pseudo-syllabic paradigm for improving the efficiency of typing in languages with transparent orthography (i.e., languages with highly regular correspondence of orthography and phonetics). By adopting a novel orthogonal framework, keyboards are defined as 2-D regular arrays of keys. Non-expert users can fast and intuitively input any possible combination of pseudo-syllables, which are text entry units with simpler consonant-vowel phonemic structure. Moreover, it is possible to input single characters in the typical letter-by-letter way. For transparent languages such as Italian and Spanish, the performed tests have shown a significant improvement in the efficiency of typing texts.

Keywords: text entry; pseudo-syllables; orthogonal keyboard; transparent orthography.

International Journal of Human-Computer Studies, Vol. 64, N. 5, May 2006, pp. 475-488
1 Introduction

The increasing diffusion of computers to almost all people living in developed countries calls for adequate fluency in typing texts to exploit the communication and service access utilities provided by Internet, and the word processing facilities provided by modern PCs. Basically, fluency in typing text requires the capability to introduce alphabetical texts in a fast way, so that the user is able to score high CPS (characters per second) or high WPM (words per minute). This can be obtained by using standard or special full keyboards, which make it possible to input one or more characters for each keystroke selection. Instead, when a reduced keyboard is strictly required to meet size constraints, such as in mobile applications, more keystrokes are, on average, required to input a single character. With full keyboards, expert (or at least well-trained) users can easily achieve text entry speeds of more than 40 WPM’s. This is quite better than the figures obtained by handwriting, which typically lie between 15 and 25 WPM’s ¹ (MacKenzie and Soukoreff, 2002). However, a lot of people are only able to type on standard keyboards at less than 10 WPM’s, which correspond to 1 or less CPS.

For non-impaired users, this low performance in typing is the consequence of insufficient use of the keyboard, so that the user does not have memorised the positions of the keys on the keyboard. As a consequence, non-expert users often use only one finger to type. In any case, a full visual exploration of the keyboard is typically necessary to select each key. For a non-expert user, the time needed for visual exploration can be modelled by the Hick-Hyman law for choice selection (or reaction) time. In its basic form, the Hick-Hyman law says that $T_v = a' + b' \log_2(n)$, where $n$ is the number of the keys, $a'$ is the response time and $1/b'$ is the bandwidth (bits per second), i.e., the rate at which humans process choices. For software keyboards and continuous text entry $a' = 0$ and a lower bound for novices is given by $1/b' = 5,7$ bps, so that $T_v = 0.2 \log_2(n)$ (this means 951 msec for $n = 27$ choices) (Soukoreff, 2002).

Another time constraint is given by the time latency between the insertions of two keys. This time cannot be less than the time needed to move the cursor or the finger from the current key to the following one. This latency time $T_l$ is quite carefully characterised using the Fitts’ law model (MacKenzie, 1992; Zhai, 2004). In its basic (one-dimensional) form, the Fitts’ law says that $T_l = a + b \log_2 \left( \frac{d}{w} + 1 \right)$, where: $d$ is the distance between the two keys, $w$ is the width of the keys, and $a$, $b$ are empirically determined constants. This model has also been used for two-dimensional configurations; in this case $d$ is the Euclidean distance between the keys (Zhai

¹This also explains why handwriting stenography is much less used than typing or computerised stenography.
et al., 2005). The Fitts’ law has also led to some bivariate pointing models, such as: $T_l = a + b \log_2 \left( \frac{d}{\min(w, h)} + 1 \right)$ (MacKenzie and Buxton, 1992; Hoffmann and Sheikh, 1994) or $T_l = a + b \log_2 \left( \sqrt{\left(\frac{d}{w}\right)^2 + \eta \left(\frac{d}{h}\right)^2} + 1 \right)$ (Accot and Zhai, 2003); where $w$ and $h$ are the width and height of the keys, respectively.

For non-expert users $T_l$ is typically lower than $T_v$. This is not the case for motor-impaired users, whose low typing speeds (typically less than 1 CPS) are directly due to the handicap itself. Although extensive training can produce some improvement concerning CPS, this is only due to the acquired memorisation of the keyboard layout, which eliminates the time overhead due to visual search. Therefore, the Fitts’ law provides a suitable time model for expert motor-impaired users, with suitable values for $a$ and $b$. In other words, the Hick-Hyman law and the Fitts’ law can be used as models to estimate the time lower bounds for non-expert/non-impaired and expert/impaired users, respectively.

We can note that if the users are able to memorise the keyboard layout, then they can achieve higher typing speed by using a keyboard with an optimal character arrangement that minimises Fitts’ law times (Lesher et al., 1998; MacKenzie and Zhang, 1999). However, as seen, the above hypothesis is not true for non-expert users. The same consideration can be done for chord keyboards, in which the user inputs characters by selecting more keys concurrently (Gopher and Raij, 1988; MacKenzie and Soukoreff, 2002).

Instead, some significant improvement can be obtained by adding a set of suitable text prediction techniques during the typing process (Darragh and Witten, 1991; Rosenfeld, 1996; Matiasek et al., 2002; Ward et al., 2002). This type of improvement is incremental to any other improvement eventually achieved at the typing level. Therefore, provided that we can obtain a significant improvement in typing efficiency with the adoption of a new keyboard paradigm, its use in combination with predictive techniques will produce an even higher reduction of the number of the keys that must be selected. Therefore, the main parameter to be minimised is $KSPC$, i.e., the number of keystrokes per character, which is defined as the mean number of key selections needed to input a character (MacKenzie and Soukoreff, 2002).

Another major issue concerns the design of interfaces that are tailored, and possibly optimised, to the native languages of the users. In fact, although English is by far the most common language for communication among people all over the world, most people who interact with computers only (or almost) use their native languages. Therefore, it is important that the paradigm of interaction be not completely tailored to the specific features of a given language. In this way, the same
paradigm can be easily applied to different target languages.

In our work, we have addressed the problem of improving typing efficiency of non-expert users in desktop applications. To this aim, we have studied and implemented a method that is suitable to achieve small $KSPC$ values by allowing the user to input more characters with a single key selection. This has been done by adopting a novel orthogonal keyboard framework, with the definition of pseudo-syllables as text entry units. According to this scheme, the keyboard configuration is defined as a two-dimensional array of keys. The user can introduce, in a direct and intuitive way, any possible combination of the possible pseudo-syllables, or single characters in the classical letter-by-letter way.

In particular, the proposed orthogonal scheme provides the non-expert user with an efficient way to reach the correct $x, y$ positions of a key without the need of a complete visual search of the keyboard. This has made it possible to obtain better choice reaction times than with standard qwerty keyboards.

This novel text entry paradigm can be directly and easily applied to the graphemic structure of whichever alphabetical language with transparent orthography (i.e., with highly regular correspondence between orthography and phonetics). As most languages have this characteristic, the orthogonal paradigm is suitable to implement specific orthogonal keyboards for a high percentage of world languages. It will be shown that for two transparent languages, such as Italian and Spanish, a significant improvement in alphanumeric text entry performance can be obtained.

The paper is organised as follows. Section 2 introduces the use of syllables as basic text entry units, and describes in detail the proposed pseudo-syllabic text entry paradigm. Section 3 describes the orthogonal keyboard framework and two orthogonal keyboard maps for Italian and Spanish. In Sections 4 and 5 some experimental results are presented and discussed. Conclusions are outlined in Section 6.

2 The text entry paradigm

2.1 Introduction

In the alphabetical human languages, each word is written and read as a sequence of graphemes, $G = (g_1...g_m)$, according to some relation with the sequence of the phonemes $P = (p_1...p_m)$ that describe the word pronunciation. Each grapheme is in turn constituted by one or more alphabetical characters. To denote the phonemes, we will use the SAMPA (Speech Assessment Methods Phonetic Alphabet) phonetic alphabet (Sampa, 2003), which is a machine-readable phonetic alphabet,
initially developed for western European languages, and then extended to eastern European and
Asian languages. In this notation, each single phoneme is included between two slash symbols
'/'. To denote the pronunciation of an entire word the sequence of phonemes may be included
between two slashes, and an apostrophe is used to denote the accented syllable (for example:
/r//o//m//a/ or /'roma/).

Typically, the number of phonemes used in a given language $L$ exceeds the 26 basic letters
of the English alphabet. For this reason, it has been necessary to introduce diacritics (such as:
č $\rightarrow$ /tS/ in some Slavic languages, or ň $\rightarrow$ /J/, in Spanish) and compound graphemes,
constituted by more alphabetical letters (such as: ch $\rightarrow$ /tS/, sh $\rightarrow$ /S/, in English).

The orthography of a target language $L$ (and the language itself) is completely transparent
(or shallow) if an injective correspondence grapheme to phoneme $g \rightarrow p$ holds for all the
possible graphemes $g_i \in L$, independently on the word or the phrase context. On the other hand,
the orthography of $L$ is completely opaque (or deep) when the pronunciation of a word typically
requires the overall evaluation of the written word as a unique logographic unit.

Most real languages are not completely transparent or completely opaque; the transparency
of a language can be related to the degree of word pre-comprehension that is required in reading.
Among western languages, Italian, Spanish and German are very transparent languages, with good
grapheme to phoneme correspondences. On the other side, English is one of the most opaque
languages. This is the main reason why it is much more difficult to learn to read English, despite
its very simple and regular syntax (Seymour et al., 2003), and why so many dyslexic people have
been detected in English speaking countries (Landerl et al., 1997; Paulesu et al., 2001) .

Moreover, writing is much easier for a transparent language, because it is not necessary to spell
the unknown terms during dictation. In fact, writing difficulty originated in the past some propos-
als for a radical reform of the English orthography to reduce the gap between spoken and written
language. Although utopian, these proposals are symptoms of the problems that are inherent in
learning and using an opaque language .

---

2 Apparently, language transparency is the main reason why this problem is only partially estimated in countries like
Italy and Spain, where many mildly dyslexic people are finally able to read correctly, although slowly.

3 However, a new English orthography would produce other problems, such as: 1) the discontinuity with all the
English literature of the past, and 2) the ambiguity the high number of English homophones would produce in a written
The advantage of a transparent orthography is even more evident after having considered that a word is composed by syllables, which constitute the basic prosodic units of the sound and holds important phonetic features such as stress, tone, and pitch. A syllabic sound is constituted by one vowel phoneme and by two, optional, beginning and ending sequences of (semi-)consonant phonemes. Although each language has its own set of syllables, only in the transparent languages it is possible to directly translate a sequence of phonemes into a unique syllable, and vice versa, without the need to capture the overall word meaning. As our aim is to define pseudo-syllables as basic units for text entry, let us briefly review the definitions of syllable. Concerning the syllable structure there are four basic theories: onset-rhyme, flat, body-coda, moraic (Kessler and Treiman, 1997).

According to the most cited onset-rhyme model, a syllable $S$ can be divided into two basic parts: onset $O$ (possibly empty) and rhyme $R$; $R$ is in turn composed by a nucleus $N$, possibly followed by a coda $C$. When the syllable starts with a consonant or semi-consonant sound, $O$ is the beginning sound of the syllable. The nucleus $N$ is constituted by the vowel sound on which the syllable sound is built 4. When the syllable ends with a consonant or semi-consonant sound, $C$ does represent this ending sound. In the absence of $C$ we have an open syllable. Instead, in the flat model a syllable is outlined as a sequence of (semi-)consonants ($C$) and vowels ($V$).

The case $CV$, which corresponds to an atomic $O$ followed by an atomic $R$ without any $C$, is by far the more common way to assemble syllables in most world languages. For example, almost all the syllables of Japanese have this structure, so that the two Japanese syllabaries, katakana and hiragana, have only 46 symbols (Baldwin et al., 2002). Also in this case English is a notable exception 5. In fact, according to a statistics of American English syllables (Wu, 1998), the eight most used syllable structures (which cover about 84% of the syllables frequencies) are:

- /CVV/ (21.19 %)
- /CVC/ (19.75 %)
- /CVCC/ (9.99 %)
- /CV/ (9.51 %)
- /VC/ (9.14 %)
- /VV/ (6.98 %)
- /CVCC/ (3.99 %)
- /VCC/ (3.85 %)

Moreover, the same statistics shows that the American English Switchboard corpus contains about 6000 different syllables. Instead, in transparent languages, such as Italian and Spanish, the correspondences between the graphemes and the phonemes are unambiguous, and much less syllables are needed to obtain an adequate text coverage. For example, the PD/DPSS Italian database text.

4 In some languages (such as English) the liquid (/r/, /l/), nasal (/m/, /n/), and velar nasal (/ng/) consonants can be the nucleus of a syllable.

5 English longest syllable is scrounged (whose graphemic and phonemic structures are $CCCVVCCVC$ and $/CCCVVCCC/$, respectively).
(Stella and Job, 2001), contains 26,870,440 syllables occurrences and only 2719 different syllables. Tables 1 and 2 show the statistics concerning the most frequent syllables and syllable structures (obviously, we have discarded all the non-word text entities). In Spanish, the frequencies of the syllable structures are roughly similar; Table 3 shows the statistics of a Spanish database with 498,464 syllables occurrences and 1148 different syllables (Justicia et al., 1996).

<table>
<thead>
<tr>
<th>N</th>
<th>SYL</th>
<th>%SYL</th>
<th>#SYL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>di</td>
<td>3.146300</td>
<td>845418</td>
</tr>
<tr>
<td>2</td>
<td>to</td>
<td>2.543800</td>
<td>683526</td>
</tr>
<tr>
<td>3</td>
<td>la</td>
<td>2.475500</td>
<td>665182</td>
</tr>
<tr>
<td>4</td>
<td>e</td>
<td>2.375700</td>
<td>638367</td>
</tr>
<tr>
<td>5</td>
<td>ti</td>
<td>2.218000</td>
<td>595974</td>
</tr>
<tr>
<td>6</td>
<td>re</td>
<td>2.043200</td>
<td>549009</td>
</tr>
<tr>
<td>7</td>
<td>a</td>
<td>1.964800</td>
<td>527947</td>
</tr>
<tr>
<td>8</td>
<td>te</td>
<td>1.816000</td>
<td>487972</td>
</tr>
<tr>
<td>9</td>
<td>le</td>
<td>1.793000</td>
<td>481781</td>
</tr>
<tr>
<td>10</td>
<td>no</td>
<td>1.708000</td>
<td>461205</td>
</tr>
<tr>
<td>11</td>
<td>ta</td>
<td>1.716400</td>
<td>461205</td>
</tr>
<tr>
<td>12</td>
<td>si</td>
<td>1.708800</td>
<td>459169</td>
</tr>
<tr>
<td>13</td>
<td>ne</td>
<td>1.622000</td>
<td>435826</td>
</tr>
<tr>
<td>14</td>
<td>ri</td>
<td>1.504500</td>
<td>404268</td>
</tr>
<tr>
<td>15</td>
<td>in</td>
<td>1.497100</td>
<td>402265</td>
</tr>
<tr>
<td>16</td>
<td>na</td>
<td>1.368400</td>
<td>367700</td>
</tr>
<tr>
<td>17</td>
<td>co</td>
<td>1.341400</td>
<td>360433</td>
</tr>
<tr>
<td>18</td>
<td>che</td>
<td>1.286500</td>
<td>345680</td>
</tr>
<tr>
<td>19</td>
<td>ra</td>
<td>1.219500</td>
<td>327683</td>
</tr>
<tr>
<td>20</td>
<td>ni</td>
<td>1.211300</td>
<td>325484</td>
</tr>
<tr>
<td>21</td>
<td>so</td>
<td>1.191400</td>
<td>320124</td>
</tr>
<tr>
<td>22</td>
<td>ca</td>
<td>1.186100</td>
<td>318722</td>
</tr>
<tr>
<td>23</td>
<td>del</td>
<td>1.142000</td>
<td>306869</td>
</tr>
<tr>
<td>24</td>
<td>li</td>
<td>1.128100</td>
<td>303124</td>
</tr>
<tr>
<td>25</td>
<td>ma</td>
<td>1.053600</td>
<td>283109</td>
</tr>
</tbody>
</table>

Table 1 The most frequent syllables in Italian

<table>
<thead>
<tr>
<th>N</th>
<th>Type</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CV</td>
<td>53.885933</td>
</tr>
<tr>
<td>2</td>
<td>CVC</td>
<td>14.882179</td>
</tr>
<tr>
<td>3</td>
<td>CCV</td>
<td>7.625462</td>
</tr>
<tr>
<td>4</td>
<td>V</td>
<td>7.116852</td>
</tr>
<tr>
<td>5</td>
<td>VC</td>
<td>6.558232</td>
</tr>
<tr>
<td>6</td>
<td>CVV</td>
<td>5.446469</td>
</tr>
<tr>
<td>7</td>
<td>CCVC</td>
<td>1.624440</td>
</tr>
<tr>
<td>8</td>
<td>CVVC</td>
<td>1.036989</td>
</tr>
<tr>
<td>9</td>
<td>CCVV</td>
<td>0.682329</td>
</tr>
<tr>
<td>10</td>
<td>CCCV</td>
<td>0.416184</td>
</tr>
<tr>
<td>11</td>
<td>VV</td>
<td>0.340491</td>
</tr>
<tr>
<td>12</td>
<td>CCCVC</td>
<td>0.144502</td>
</tr>
</tbody>
</table>

Table 2 Frequencies of the syllable structures in Italian
2.2 The pseudo-syllabic text entry paradigm

Now, let us define the general criteria for selecting the syllabic text entry units to be used in the keyboard, and placing them in the keyboard layout. The following criteria are outlined in decreasing order of importance.

C1 The user should be able to identify the position of the basic keyboard keys (i.e., the keys selected without using any additional shift key) without performing a complete visual exploration of the keyboard and without the need to memorise the entire keyboard map. Therefore, the allocation of the text units to the basic keyboard keys should be highly ordered, and the horizontal and vertical positions of a basic key should be easily inferable from the graphemic structure of the related syllabic entity. It is worth noting that the adoption of an ordered keyboard arrangement greatly improve the efficiency of a text entry operation performed under conscious control by a motor-impaired user, which has a much reduced ability to automatise movements. This is the reason why the alphabetically sorted keyboard is so widely used by motor-impaired children.

C2 The above criterion should also be adopted for the allocation of the text units to the composed keyboard keys, i.e., the keys selected with the use of one of the available shift keys.

C3 The most frequently occurring text entry units (dominant text entry units) should be directly assigned to the basic keyboard keys. In this way, only one gesture is needed to input a dominant text entry unit.

C4 The other significant text entry units (subordinate text entry units) should be assigned to composed keyboard keys. In this way, two (but possibly concurrent) gestures are needed, involving the use of one of the shift keys.

<table>
<thead>
<tr>
<th>N</th>
<th>Type</th>
<th>%</th>
<th>N</th>
<th>Type</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CV</td>
<td>53.37</td>
<td>9</td>
<td>CVV</td>
<td>0.07</td>
</tr>
<tr>
<td>2</td>
<td>CVC</td>
<td>20.41</td>
<td>10</td>
<td>VV</td>
<td>0.05</td>
</tr>
<tr>
<td>3</td>
<td>V</td>
<td>7.79</td>
<td>11</td>
<td>CVVC</td>
<td>0.05</td>
</tr>
<tr>
<td>4</td>
<td>VC</td>
<td>6.52</td>
<td>12</td>
<td>CCV</td>
<td>0.04</td>
</tr>
<tr>
<td>5</td>
<td>CVV</td>
<td>4.93</td>
<td>13</td>
<td>CCVV</td>
<td>0.02</td>
</tr>
<tr>
<td>6</td>
<td>CCV</td>
<td>3.48</td>
<td>14</td>
<td>CVCC</td>
<td>0.02</td>
</tr>
<tr>
<td>7</td>
<td>CVVC</td>
<td>2.10</td>
<td>15</td>
<td>VVC</td>
<td>0.01</td>
</tr>
<tr>
<td>8</td>
<td>CCVC</td>
<td>1.13</td>
<td>16</td>
<td>VCC</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Table 3 Frequencies of the syllable structures in Spanish
C5 The rarely occurring text entry units should not be directly allocated to the keyboard. They should be inputted by selecting a sequence of basic or composed keys.

For western languages, it is not possible to meet the above criteria by simply adopting the whole syllable set as text entry units, although at first sight this would seem the ideal choice. In fact, from the statistics outlined in Section 2.1 it is apparent that: 1) no keyboard can contain the set of the most frequent syllables, and 2) the ordering required by criteria C1 and C2 is not compatible with the frequencies of the syllable structures.

Therefore, we have studied a text entry paradigm in which the keyboard array contains a set of pseudo-syllables (or p-syllables). This term had already been used in the field of speech analysis and recognition to denote simple phonemic sequences (typically restricted to /C//V/) (Garcia et al., 1995; De Mori and Galler, 1996; Farinas et al., 2002). In our work, we have introduced a novel definition of p-syllable, to match as much as possible the above criteria and to maintain a simple and efficient graphemic structure. Starting from the statistics outlined in Tables 1 to 3, which show the marginal frequencies of the different syllable structures, the following considerations have been taken into account.

1. The most common syllable structure has the graphemic sequence CV, with marginal frequencies exceeding 50%. This percentage is even higher at the phonemic level. In fact, in many syllables with phonemic structure /C//V/, the consonant or the vowel sounds are denoted by graphemes with two or more letters. For example, in many languages a double consonant grapheme denotes a single stressed consonant phoneme or a double vowel grapheme denotes a single stressed consonant phoneme. Therefore, the p-syllable set should include at least a subset of the graphemic sequences $bC$ belonging to the language, where we denote by $bC$ a generic consonant grapheme corresponding to a consonant phoneme, and by $bV$ a generic vowel grapheme corresponding to a vowel phoneme.

2. There is a significant percentage of syllables with the graphemic structure $bV$. Therefore, the p-syllable set should include at least a subset of the graphemic sequences $bV$ belonging to the language.

3. The p-syllable set should include the simple graphemic sequences $bV$ and $bC$. This choice is based on the high frequencies shown by single vowel syllables, and on the need to type the syllables that are not included in the p-syllable set.
According to the above points, we can start by introducing the \textit{p-syllable set} for a language \( L \): \( PS_L \subseteq \{ \hat{C}[\hat{V}] \mid \hat{V}[\hat{C}] \} \), where \( \hat{C}, \hat{V} \) are all the possible consonant and vowel graphemes belonging to \( L \). The following step is to define the p-syllable set in a more precise way.

First, let \( CS'_L = \{ \forall C_i \in L \} \) (\textit{consonant set}) be the set of all the usual consonants \( C_i \) used to write texts in the language \( L \), and let \( VW'_L = \{ \forall V_i \in L \} \) (\textit{vowel set}) be the set of all the usual vowels \( V_i \) used to write texts in the language \( L \). For usual letters we intend those strictly belonging to the orthography of the language or used to write frequently used foreign words. For example, the five letters \( j, k, w, x, y \) do not strictly belong to the Italian alphabet, but are used to write a large number of foreign words (typically English). Instead, the use of other letters with diacritics, e.g.: \( \ddot{o}, \ddot{e} \) is very infrequent. A good rule is to consider as usual letters of the language \( L \) only those used in the standard keyboard configuration for \( L \). This means that, in general, \( CS'_L \) and \( VW'_L \) can contain letters with accents or diacritics. Moreover, let \( CS_L = CS'_L \cup \varepsilon \) and \( VW_L = VW'_L \cup \varepsilon \) be the \textit{extended consonant set} and the \textit{extended vowel sets}, i.e., extended with the empty grapheme \( \varepsilon \). Let \( N_{CS_L} = |CS_L| \) and \( N_{VW_L} = |VW_L| \) be the cardinalities of the two sets.

Now, let us extend the above definitions to include the graphemes that are constituted by two or more alphabetical letters. Let \( CS''_L = \{ \forall \hat{C}_i \in L \} \) (\textit{consonant graphemic set}) be the set of all the usual consonant graphemes \( \hat{C}_i \) used to write texts in the language \( L \), and let \( VW''_L = \{ \forall \hat{V}_i \in L \} \) (\textit{vowel graphemic set}) be the set of all the usual vowel graphemes \( \hat{V}_i \) used to write texts in the language \( L \). This means that \( \hat{C}_i \) could contain multiletter consonants, such as \( tt \) or \( ch \), and \( \hat{V}_i \) could contain multiletter vowels such as \( aa \) or \( oe \). Moreover, let \( CS_L = CS''_L \cup \varepsilon \) and \( VW_L = VW''_L \cup \varepsilon \) be the \textit{extended consonant graphemic set} and the \textit{extended vowel graphemic set}, where \( \varepsilon \) is the empty grapheme. Let \( N_{CS_L} = |CS_L| \) and \( N_{VW_L} = |VW_L| \) be the cardinalities of the two sets. We can now define more precisely the p-syllable set:

\[
PSF_L = \{ \forall P_i = \hat{C}_i\hat{V}_i \in L, \hat{C}_i \in CS_L, \hat{V}_i \in VW_L \} \quad \text{(forward p-syllable set)}
\]
\[
PSR_L = \{ \forall P_i = \hat{V}_i\hat{C}_i \in L, \hat{V}_i \in CS_L, \hat{C}_i \in VW_L \} \quad \text{(reverse p-syllable set)}
\]
\[
PS_L = PSF_L \cup PSR_L \quad \text{(complete p-syllable set)}
\]
\[
N_{PS_L} = |PS_L| \quad \text{(cardinality of the p-syllable set)}
\]

The use of \( PS_L \) makes it possible to input text entry units very close to the syllabic structure of the words to be inserted. In fact, in most cases a p-syllable constitutes a syllable or at least a part of it. For example, when the syllable contain an onset with two different consonants, they
are subdivided into two different adjacent p-syllables. The only notable exception to this rule is given by syllables that contain a double consonant grapheme, when the orthographic rules of the language require the subdivision of the grapheme into two adjacent syllables. For example, the Italian word \textit{fat} − \textit{to} ( = fact) has two syllables (\textit{fat} and \textit{to}) and two different p-syllables (\textit{fa} and \textit{tto}). However, this subdivision rule is somewhat conventional; in fact, not all the linguists involved in the final fixation of Italian orthography at the end of the XIX century agreed on the above rule (Migliorini, 1991). Therefore, although not conventional, the p-syllables definition matches well the perception of a double consonant grapheme as a single phoneme.

3 Pseudo-syllabic keyboards

3.1 The orthogonal keyboard scheme

As stated before, the p-syllables should be assigned to the keyboard keys in an ordered way, so that the user can select a key without the need to memorise the keyboard map. This is very important because of the high number of keys required for a p-syllabic keyboard. In fact, an alphabetical keyboard only requires 26 keys to introduce all the English alphabetical characters, and this number slightly exceed 30 for languages with accented letters and diacritics (we will denote by \(N_L\) the cardinality of the alphabetical set for the language \(L\)). In this context, an expert user can memorise the keyboard map and a non-expert user can visually explore the keyboard configuration in a reasonable time.\(^6\) Instead, \(N_{PS_L}\) is typically at least one order of magnitude larger than \(N_L\). This means that, in absence of a suitable ordering for keys allocation, the user should memorise some hundreds positions for fast accessing all the possible p-syllables in the keyboard. This would be a difficult task even for expert users. Let us denote by \(N_A\) the size of a p-syllabic keyboard, and by \(N_K\) the number of shift keys used to select the composed keys. \(N_A\) will depend on the number of basic keys directly associated to p-syllables. Therefore, taking into account that the keyboard must also contain some non-alphabetical and control characters, we have that \(N_A \geq N_{PS_L}/N_K\).

The ordering strategy that will be now described allows the non-expert user to fast access any possible p-syllable in the keyboard, with a limited cognitive load. It is worth noting that this model is valid for any alphabetical language \(L\), although the provided examples will concern \(L = \text{i}t \mid \text{s}p\).

\(^6\)It is worth noting that the standard keyboards, with the notable exception of the \textit{ABCDEF} one, are characterised by a non-ordered arrangement of the alphabetical keys.
Let $K(x, y), \ x \in [1, N_x], y \in [1, N_y]$ be the rectangular keyboard array onto which the p-syllables have to be mapped (Figure 1). The ordering of the keyboard array is done by adopting the following ordering rule:

The keyboard array is built by allocating to the same column all the keys with a given consonant grapheme, and by allocating to the same row all the keys with a given vowel grapheme.

For example, the p-syllables $ga$ and $ge$ would be associated to the same keyboard column because both contain the same consonant grapheme $g$, whereas they have two different vowel graphemes. Instead, the p-syllables $ca$ and $ta$ would be associated to the same keyboard row because both contain the same vowel grapheme $a$. It is worth noting that also the p-syllables $ta$ and $t\hat{a}$ would be associated to different keyboard rows because they contain two different vowel graphemes. The formal definition of the keyboard layout is done through two mapping functions.

To specify $F_{PL}$ and $R_{PL}$, let us define two mapping functions that univocally and independently provide two indexes, which are related to the consonant and the vowel grapheme, respectively.

$F_{PL} : PSF_L \rightarrow [1, N_x] \times [1, N_y] \in \mathbb{N}^2$ (forward mapping function)

$R_{PL} : PSR_L \rightarrow [1, N_x] \times [1, N_y] \in \mathbb{N}^2$ (reverse mapping function)

Since the domains of $f_c$ and $f_v$ contain the empty grapheme, the keyboard array also contains all the p-syllables composed by a single $\hat{C}$ or $\hat{V}$ grapheme. In this way it is guaranteed the graphemic completeness. Finally, the ordering rule is met by imposing that:
Therefore, $F_{PL}(p)$ and $R_{PL}(p)$ together constitute the keyboard mapping function, which maps $2 \times (N_x \times N_y)$ possible p-syllables to the $(N_x \times N_y)$ keyboard array. In fact, these relations implicate that two p-syllables with the same consonant and vowel graphemes are mapped to the same key of the keyboard. The selection between the two p-syllables, which belong to $PSF_L$ and $PSR_L$ respectively, can be done by using a specific forward/reverse shift (FR-Sh). The choice of the default selection (i.e., with FR-Sh=off) would be in principle free. However, as the marginal frequencies of the p-syllables belonging to $PSF_L$ are significantly higher than those of p-syllables belonging to $PSR_L$, we have chosen the following association:

$$K(x, y) \rightarrow PSF_L(p) = (x, y) \quad (FR-Sh = \text{off})$$
$$K(x, y) \rightarrow PSR_L(p) = (x, y) \quad (FR-Sh = \text{on})$$

The proposed keyboard arrangement (orthogonal keyboard) satisfies the required ordering criterion. In fact, a key is reached by orthogonally and separately applying the access rules for the consonant ($f_c$) and vowel ($f_v$) graphemes that constitute the p-syllable. (Curatelli and Martinengo, 2004). This means that the user can deduce the horizontal position of the p-syllable by simply analysing its consonant grapheme, and the vertical position of the p-syllable by simply analysing its vowel grapheme. Obviously the keyboard mapping function could implement the opposite choice, by assigning vowels to x and consonants to y. However, it is not practical to have a keyboard that is more extended vertically (because $|CS_L| > |VW_L|$).

Now, let us consider the ordering given by the $f_c$ and $f_v$ functions. Since our aim is to provide an intuitive way to memorise the key positions, the rows and columns are built in lexicographic order according to the first letter of each grapheme. In this way, the user has a well-known and quick way to guess the horizontal and vertical positions of the key. To provide a compact definition with respect to the extended sets $CS_L$ and $VW_L$, it is necessary to define a lexicographic order for the empty grapheme $\varepsilon$. Our choice has been to impose it as the smallest element for both the horizontal and vertical orderings:

$$f_c(\varepsilon) < f_c(\tilde{C}_i), f_v(\varepsilon) < f_v(\tilde{V}_j) \quad \forall \tilde{C}_i \in \overline{CS_L}, \forall \tilde{V}_j \in \overline{VW}_L$$

In this way, the ordering is defined as follows:

$$f_c(C_i) < f_c(C_j) \iff \tilde{C}_i < \tilde{C}_j \quad \forall \tilde{C}_i, \tilde{C}_j \in \overline{CS_L}$$
$$f_v(V_k) < f_v(V_l) \iff \tilde{V}_k < \tilde{V}_l \quad \forall \tilde{V}_k, \tilde{V}_l \in \overline{VW}_L$$
The novelty of the proposed scheme lies in the fact that the access to the position of a syllabic-like key is cognitively split into two separate accesses, through the consonant and vowel phonemes of the syllable. To evaluate the advantage of the proposed method, let us start by considering the number of keyboard positions expert users have to memorise for fast access. Let us consider a (hypothetical) completely transparent language \( \tilde{L} \) such that \( N_{CS_{\tilde{L}}} = N_{CS_L} \) and \( N_{VW_{\tilde{L}}} = N_{VW_L} \).

According to the proposed scheme, the number of different positions to be memorised is equal to \( NK_O = (N_x + N_y) \). This number is comparable to the number of different alphabetical keys for a standard keyboard, i.e., \( (N_x + N_y - 2) \). Instead, in the case of a completely non-ordered arrangement of the keyboard keys, \( NK_{NO} = 2 \times (N_x - 1) \times (N_y - 1) + N_x + N_y - 2 \) p-syllable to key associations should be memorised. As an example, let us define this hypothetical completely transparent and orthographically simple language \( \tilde{L} \), with a one-to-one correspondence between each single-letter grapheme and its phoneme:

\[
\begin{align*}
\overline{CS}_{\tilde{L}} &= CS_L = \{\varepsilon, b, c, d, f, g, h, j, k, l, m, n, p, q, r, s, t, v, w, x, y, z\} \\
\overline{VW}_{\tilde{L}} &= VW_L = \{a, e, i, o, u\} \\
N_{\overline{CS}_{\tilde{L}}} &= N_{CS_L} = 22, \quad N_{\overline{VW}_{\tilde{L}}} = N_{VW_L} = 6, \quad N_x = 22, \quad N_y = 6
\end{align*}
\]

In this simple case, the positions to be memorised would be \( NK_{NO} = 236 \) vs. \( NK_O = 28 \). The difference between the non-ordered and the orthogonal p-syllable keyboard arrangement is even more evident when increasing \( N_x \) and \( N_y \) (see Table 4). Therefore, using the proposed orthogonal scheme, we can design a highly modular p-syllable keyboard (H-Keyb) for the language \( \tilde{L} \) (Figure 2).

<table>
<thead>
<tr>
<th>( N_x )</th>
<th>( N_y )</th>
<th>( NK_O )</th>
<th>( NK_{NO} )</th>
<th>( \frac{NK_{NO}}{NK_O} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>6</td>
<td>28</td>
<td>236</td>
<td>8.43</td>
</tr>
<tr>
<td>30</td>
<td>6</td>
<td>36</td>
<td>324</td>
<td>9.00</td>
</tr>
<tr>
<td>42</td>
<td>12</td>
<td>54</td>
<td>952</td>
<td>17.63</td>
</tr>
<tr>
<td>48</td>
<td>12</td>
<td>60</td>
<td>1092</td>
<td>18.20</td>
</tr>
</tbody>
</table>

**Table 4:** No of the positions to be memorised for orthogonal and non-ordered keyboards

Now, let us evaluate the advantage of the proposed method for non-expert users. For they have not memorised the keyboard positions, the non-expert users need to visually search the keys. In this case, according to the Hick-Hyman law for visual search time (Soukoreff, 2002), we have that \( T_v = b' \log_2(NK_O) \) instead of \( T_v = b' \log_2(NK_{NO}) \). Therefore, with the orthogonal scheme also non-expert users are able to score quite lower access times with respect to a corresponding random p-syllable layout.
It is worth noting that our approach is quite different from the typical approach found in literature, which is based on the selection of more single-letter keys to build a syllabic entity. For example, Joshi et al., 2004 describes a keyboard designed for Devanagari script, which is used for many Indian languages, such as hindi (Devanagari is a so-called abugida alphabet, in which each consonant has an inherent a vowel; the default vowel can be changed by adding a different vowel sign, which slightly changes the consonant shape). To produce a syllabic grapheme, the user must type all the different consonant (selected among 25), semi-vowel (9), and vowel (9) keys that phonetically denote the syllable, together with some additional selection keys. Therefore, although the different types of phonetic keys are cleverly arranged in separate blocks to ease the cognitive effort, more key pressures (up to four) are typically required to input a single syllabic symbol (in fact, only the $C - V$ structure with $V = a$ requires one single key).

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td>f</td>
<td>g</td>
<td>h</td>
<td>j</td>
<td>k</td>
<td>l</td>
<td>m</td>
<td>n</td>
<td>p</td>
<td>q</td>
<td>r</td>
<td>s</td>
<td>t</td>
<td>v</td>
<td>w</td>
<td>x</td>
<td>y</td>
<td>z</td>
</tr>
<tr>
<td>2</td>
<td>a</td>
<td>b</td>
<td>a</td>
<td>d</td>
<td>e</td>
<td>a</td>
<td>g</td>
<td>a</td>
<td>h</td>
<td>a</td>
<td>j</td>
<td>k</td>
<td>a</td>
<td>l</td>
<td>m</td>
<td>a</td>
<td>n</td>
<td>p</td>
<td>q</td>
<td>r</td>
<td>s</td>
</tr>
<tr>
<td>3</td>
<td>e</td>
<td>b</td>
<td>c</td>
<td>e</td>
<td>d</td>
<td>f</td>
<td>e</td>
<td>g</td>
<td>h</td>
<td>j</td>
<td>k</td>
<td>l</td>
<td>m</td>
<td>n</td>
<td>p</td>
<td>q</td>
<td>r</td>
<td>s</td>
<td>t</td>
<td>v</td>
<td>w</td>
</tr>
<tr>
<td>4</td>
<td>i</td>
<td>b</td>
<td>c</td>
<td>i</td>
<td>d</td>
<td>f</td>
<td>i</td>
<td>g</td>
<td>h</td>
<td>i</td>
<td>j</td>
<td>k</td>
<td>l</td>
<td>m</td>
<td>n</td>
<td>p</td>
<td>q</td>
<td>r</td>
<td>s</td>
<td>t</td>
<td>v</td>
</tr>
<tr>
<td>5</td>
<td>o</td>
<td>b</td>
<td>o</td>
<td>d</td>
<td>f</td>
<td>g</td>
<td>o</td>
<td>h</td>
<td>j</td>
<td>k</td>
<td>l</td>
<td>m</td>
<td>n</td>
<td>p</td>
<td>q</td>
<td>r</td>
<td>s</td>
<td>t</td>
<td>v</td>
<td>w</td>
<td>x</td>
</tr>
<tr>
<td>6</td>
<td>u</td>
<td>b</td>
<td>u</td>
<td>d</td>
<td>f</td>
<td>g</td>
<td>u</td>
<td>h</td>
<td>j</td>
<td>k</td>
<td>l</td>
<td>m</td>
<td>n</td>
<td>p</td>
<td>q</td>
<td>r</td>
<td>s</td>
<td>t</td>
<td>v</td>
<td>w</td>
<td>x</td>
</tr>
</tbody>
</table>

(a) FR-Sh = off

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td>f</td>
<td>g</td>
<td>h</td>
<td>j</td>
<td>k</td>
<td>l</td>
<td>m</td>
<td>n</td>
<td>p</td>
<td>q</td>
<td>r</td>
<td>s</td>
<td>t</td>
<td>v</td>
<td>w</td>
<td>x</td>
<td>y</td>
<td>z</td>
</tr>
<tr>
<td>2</td>
<td>a</td>
<td>b</td>
<td>a</td>
<td>d</td>
<td>e</td>
<td>a</td>
<td>g</td>
<td>a</td>
<td>h</td>
<td>a</td>
<td>j</td>
<td>k</td>
<td>a</td>
<td>l</td>
<td>m</td>
<td>a</td>
<td>n</td>
<td>p</td>
<td>q</td>
<td>r</td>
<td>s</td>
</tr>
<tr>
<td>3</td>
<td>e</td>
<td>b</td>
<td>c</td>
<td>e</td>
<td>d</td>
<td>f</td>
<td>e</td>
<td>g</td>
<td>h</td>
<td>j</td>
<td>k</td>
<td>l</td>
<td>m</td>
<td>n</td>
<td>p</td>
<td>q</td>
<td>r</td>
<td>s</td>
<td>t</td>
<td>v</td>
<td>w</td>
</tr>
<tr>
<td>4</td>
<td>i</td>
<td>b</td>
<td>c</td>
<td>i</td>
<td>d</td>
<td>f</td>
<td>i</td>
<td>g</td>
<td>h</td>
<td>i</td>
<td>j</td>
<td>k</td>
<td>l</td>
<td>m</td>
<td>n</td>
<td>p</td>
<td>q</td>
<td>r</td>
<td>s</td>
<td>t</td>
<td>v</td>
</tr>
<tr>
<td>5</td>
<td>o</td>
<td>b</td>
<td>o</td>
<td>d</td>
<td>f</td>
<td>g</td>
<td>o</td>
<td>h</td>
<td>j</td>
<td>k</td>
<td>l</td>
<td>m</td>
<td>n</td>
<td>p</td>
<td>q</td>
<td>r</td>
<td>s</td>
<td>t</td>
<td>v</td>
<td>w</td>
<td>x</td>
</tr>
<tr>
<td>6</td>
<td>u</td>
<td>b</td>
<td>u</td>
<td>d</td>
<td>f</td>
<td>g</td>
<td>u</td>
<td>h</td>
<td>j</td>
<td>k</td>
<td>l</td>
<td>m</td>
<td>n</td>
<td>p</td>
<td>q</td>
<td>r</td>
<td>s</td>
<td>t</td>
<td>v</td>
<td>w</td>
<td>x</td>
</tr>
</tbody>
</table>

(b) FR-Sh = on

Figure 2: The orthogonal keyboard map for the language $\tilde{L}$ (H-Keyb).

### 3.2 Optimised orthogonal keyboards

The orthogonal arrangement above proposed for the language $\tilde{L}$ requires a keyboard with $22 \times 6$ p-syllabic keys. Moreover, other keys have to be added for selecting non-usual characters and control characters. Instead, a real transparent languages (such as Italian or Spanish) would require much larger orthogonal keyboards. However, this fact is typically unacceptable because:
1. The keyboard would occupy too much space on the desk or too much screen area (for a soft keyboard).

2. The larger size would intrinsically require more time for moving the finger or the cursor from a key to the next one. This derives directly from the application of the Fitts’ law (Accot and Zhai, 2003). In fact, selecting one key at the time is typical for motor-impaired and non-expert users working with a hardware keyboard, and is the rule for all the users working with pen pointing devices or with soft keyboards.

3. The access time would be slower because of the need to visually discriminate the target key in a larger array.

Therefore, there is the need to reduce the size of the orthogonal array, at the same time preserving the orthogonal arrangement, which is the core of the access paradigm proposed. This can been obtained by applying one or more of the following steps to generate an optimised orthogonal keyboard.

- **Column deletion (CD)** - It consists in the elimination of an entire column whose p-syllables have low marginal frequencies in the target language.

- **Row deletion (RD)** - It consists in the elimination of an entire row whose p-syllables have low marginal frequencies in the target language.

- **Column folding (CF)** - If all the keys that belong to a given column contain p-syllables with low marginal frequencies, the column is folded with another column. The selection of its (composed) keys is done by selecting a specific CF-Sh key (unique for all the folded columns).

- **Row folding (RF)** - If all the keys that belong to a row contain p-syllables with low marginal frequencies, the row is folded with another row. The selection of its (composed) keys is done by selecting a specific RF-Sh key (unique for all the folded rows).

- **Column merging (CM)** - Let \( S_i \) and \( S_j \) be two subsets of the keys row indexes that belong to the \( i-th \) and \( j-th \) columns respectively (\( 1 \leq i, j \leq N_{\bar{C}_S L} \)), and that refer to p-syllables with zero or low marginal frequencies. Let \( \bar{S}_i = \{1,...,N_{\bar{C}_S L}\} - S_i \) and \( \bar{S}_j = \{1,...,N_{\bar{C}_S L}\} - S_j \) be the subsets of the keys row indexes belonging to the same columns and referring to p-syllables with significant marginal frequencies. Therefore, if \( \bar{S}_i \cap \bar{S}_j = \emptyset \), it is possible to merge all the keys that are identified by subsets \( \bar{S}_i \) and \( \bar{S}_j \) to only one column (either \( i \) or \( j \)).
The optimisation steps described above can be suitably used to produce keyboards with reduced size. This can be done either manually or automatically, with the aid of a specific program. In the first case, the user decides, according to his/her language knowledge, which p-syllables have to be eliminated from the dominant p-syllable set. In the second case, the choice is done by a program using the language statistics of the p-syllables frequencies, which have been extracted from some significant corpus. Although in this paper we have applied only manual optimisations, automatic optimisation is a promising research field for the future.

A possible criticism to our approach is that, even after the optimizations, the keyboard would still contain more keys than any current state-of-the-art keyboard. This would clearly limit the use of the keyboard, especially since computing is becoming more and more mobile, and surface space is limited in mobile devices. In fact, this is certainly true for mobile applications. However, for desktop applications, at both home and school, this is not a significant problem, because a larger hardware keyboard can be anyway placed on the desk and a larger software keyboard can be displayed on the monitor screen.

### 3.3 Italian orthogonal keyboard

Italian is well suited for a realistic case study on the application of our methodology; in fact, it is a transparent language with a high number of different graphemes (mainly at the consonant level). In particular, almost all the single consonants can be doubled to denote the corresponding stressed phonemes. The basic (i.e., non-optimised) orthogonal keyboard can be built by taking into account the set of consonant and vowel graphemes in Italian, extended with the 5 non-native, but common, letters \( j, k, w, x, y \). In Italian, an accent is used only on the final vowel of a word (when the tonic syllable is the final one) and in a few monosyllabic words \(^7\). The accent can be grave or acute for \( e \) and only grave for the other vowels: \( à, ì, ò, ù \). The consonant and vowel graphemes are defined as:

\[
\begin{align*}
\overline{CS}_{it} &= \{\varepsilon, b, bb, c, cc, d, dd, f, ff, g, gg, l, ll, m, mm, n, nn, \\
p, pp, qu, cqu, r, rr, s, ss, t, tt, v, vv, z, zz, \\
ci, cci, chi, cchi, gi, ggi, ghi, gghi, gli, gn, sc, \tilde{h}, j, k, w, x, y\} \\
\overline{VW}_{it} &= \{\varepsilon, a, e, i, o, u, à, ì, ò, ù\}
\end{align*}
\]

\(^7\)Strictly speaking, some words that are written with the same sequence of letters should be distinguished by an explicit accent. However, this convention is typically limited to printed books and is not adopted in school or in any other use of the written language.
Therefore, $N_{CS_{it}} = 48$ and $N_{VW_{it}} = 12$, so that the complete orthogonal keyboard would require an $N_x \times N_y = 48 \times 12$ array of keys. The complete p-syllable set is defined as

\[ PS_{it} = PSF_{it} \cup PSR_{it} \]

\[ PSF_{it} = \{ \forall P_i = \widehat{C}_i \widehat{V}_i \in it, \widehat{C}_i \in CS_{it}, \widehat{V}_i \in VW_{it} \} \]

\[ PSR_{it} = \{ \forall P_i = \widehat{V}_i \widehat{C}_i \in it, \widehat{C}_i \in CS_{it}, \widehat{V}_i \in VW_{it} \} \]

In Italian, the letters $j, k, w, x, y$ (isolated or followed by a vowel) do not belong to Italian native words, so that they are by far less used in the written texts. Moreover, the typical use of the consonant $h$ is to provide, when it follows $c$ and $g$, the guttural sounds /k/ and /g/, or to be mute in some specific words, the most significant ones being four verbal forms of the present tense of *avere* (to have). Therefore, it is convenient to eliminate all the p-syllables containing these six letters by applying a sequence of CD optimisations. It is worth noting that the most used words can be anyway allocated in the non-orthogonal part of the keyboard. A second optimisation concerns the accented p-syllables, which are by far less frequent in Italian. Therefore, it is convenient to eliminate the accented p-syllables from the orthogonal array by applying a sequence of RD optimisations. Also in this case, it is anyway possible to allocate some accented p-syllables with significant frequencies to the non-orthogonal section of the keyboard. After these steps, we have that $N_x = 42$ and $N_y = 6$.

A further optimisation has been done after observing that in Italian double consonants are less frequently used than single consonants. Therefore, the keyboard size can be further reduced by performing a sequence of CF optimisations. Each p-syllable $\widehat{C}_d \widehat{V}$, where $\widehat{C}_d$ is a double consonant, is mapped onto the same position of the p-syllable $\widehat{C} \widehat{V}$, where $\widehat{C}$ is the same, but single, consonant. It is worth noting that this optimisation still satisfies the ordering rule. At the end, we have that $N_x = 23$ and $N_y = 6$. This means that the orthogonal keyboard will contain two reduced consonant and vowel sets. The final keyboard layout for dominant p-syllables (It-Keyb) is shown in Figure 3.

The complete alphabetical part of the keyboard can be obtained by extending the basic orthogonal keyboard with a 7-th row, which contains all the letters deleted during the CD and RD optimisations:

\[
7 \ldots \ldots \; à \; è \; é \; ì \; ò \; ù \; h \; j \; k \; w \; x \; y \ldots \ldots
\]

The dotted places can be freely filled with other text or control keys. For example, one could
possibly add the single q letter to directly manage the few foreign words written without the vowel u, such as Iraq or Qatar.

### 3.4 Spanish keyboard map

The basic (non optimised) Spanish orthogonal keyboard can be built by taking into account the set of consonant and vowel graphemes present in the language. In Spanish the consonant graphemic set is smaller than in Italian as the use of double consonants is almost limited to l and r. On the other hand, j, k, w, x, y are part of the language, and h can be followed by whichever vowel. Moreover, there is a high number of accented words, because an accent must be used to denote the tonic syllable every time it is not the syllable identified by the basic orthographic rules. This accent is always acute: á, é, í, ó, ù. Therefore, the consonant and vowel sets are:

\[
\begin{align*}
\overline{CS}_{sp} &= \{\varepsilon, b, c, ch, cu, d, f, g, h, j, k, l, ll, m, n, \check{n}, p, qu, r, rr, s, t, v, w, x, y, z\} \\
\overline{VW}_{sp} &= \{\varepsilon, a, e, i, o, u, á, é, í, ó, ù\}
\end{align*}
\]

Therefore, we have that \(N_{\overline{CS}_{sp}} = 27\) and \(N_{\overline{VW}_{sp}} = 11\), so that the complete orthogonal keyboard would require \(N_x \times N_y = 27 \times 11\) keys. The complete p-syllable set is defined as:

\[
\begin{align*}
\overline{PS}_{sp} &= \overline{PSF}_{sp} \cup \overline{PSR}_{sp} \\
\overline{PSF}_{sp} &= \{ \forall P_i = \hat{C}_i \hat{V}_i \in sp, \hat{C}_i \in \overline{CS}_{sp}, \hat{V}_i \in \overline{VW}_{sp} \} \\
\overline{PSR}_{sp} &= \{ \forall P_i = \hat{V}_i \hat{C}_i \in sp, \hat{C}_i \in \overline{CS}_{sp}, \hat{V}_i \in \overline{VW}_{sp} \}
\end{align*}
\]

In this case, the horizontal size of the orthogonal keyboard is acceptable. Instead, a vertical optimisation can be done after observing that, in Spanish, accented p-syllables are less frequently

---

\(^8\)It is worth noting that in Spanish ll, rr, and ch have been up to 1994 orthographically treated as single letters. In dictionaries, they were considered as specific letters and sorted after the related single consonants.
used than non-accented ones. Therefore, we have performed a sequence of RF optimisations. Each p-syllable $\tilde{C}V_a$, where $\tilde{V}_a$ is a vowel with an acute accent, is mapped onto the same position of the p-syllable $\tilde{V}$, where $\tilde{V}$ is the same but non-accented vowel. At the end, we have that $N_x = 27$ and $N_y = 6$. This means that the orthogonal keyboard will contain a reduced vowel set. The final keyboard layout for dominant p-syllables (Sp-Keyb) is shown in Figure 4. It is worth noting that it is not strictly necessary to add another row, because all the letters belonging to $\tilde{C}S_{sp}$ and $\tilde{V}W_{sp}$ are already included in the orthogonal keyboard.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 |
|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 1 | . | b | c | ch | cu | d | f | g | h | j | k | l | ll | m | n | ñ | p | qu | r | rr | s | t | v | w | x | y | z |
| 2 | a | ba | ca | cha | cu | da | fa | ga | ha | ja | ka | la | lla | ma | na | ña | pa | qua | ra | rra | sa | ta | va | wa | xa | ya | za |
| 3 | e | be | ce | che | cue | de | fe | ge | he | je | ke | le | lle | me | ne | ñe | pe | que | re | rre | se | te | ve | we | xe | ye | ze |
| 4 | i | bi | ci | chi | cui | di | fi | gi | hi | ji | ki | li | lli | mi | ni | ñi | pi | qui | ri | rri | si | ti | vi | wi | xi | yi | zi |
| 5 | o | bo | co | cho | cuo | do | fo | go | ho | jo | ko | lo | llo | mo | no | ño | po | quo | ro | rro | so | to | vo | wo | xo | yo | zo |
| 6 | u | bu | cu | chu | cuu | du | fu | gu | hu | ju | ku | lu | lli | mu | nu | ñu | pu | quu | ru | rru | su | tu | vu | su | xu | yu | zu |

FR-Sh = off, RF-Sh = off

Figure 4: Spanish orthogonal keyboard (Sp-Keyb) for dominant p-syllables.

4 Evaluation

To validate the proposed scheme, It-Keyb (extended with the 7-th row) and Sp-Keyb have been implemented, as both software and hardware keyboards. Each keyboard has been extended by also providing:

- ten keys for the digits;
- keys for the punctuation marks and separators.

4.1 Evaluation parameters

Let us introduce some parameters that will be traced during the tests or computed for the evaluation of the extended orthogonal keyboard layouts.

- $\text{word}$: total number of words;
- $\text{char} = (\text{alpha} + \text{other})$: total number of characters;
- $\text{alpha}$: total number of alphanumeric characters;
• other: total number of non-alphanumeric characters;

• space: total number of space characters;

• sh: total number of capital letter shifts;

• shsel: total number of FR, RF, and CF shifts;

• alphak: total number of alphanumeric keys typed on the keyboard;

• \( WPM = \frac{\text{[Average number of words]}}{\text{[Time in minutes]}} \): average number of words typed per minute;

• \( T_O \): total time (in seconds) to type a text using an orthogonal keyboard;

• \( T_Q \): total time (in seconds) to type a text using a qwerty keyboard;

• \( WPM_O : WPM \) for an orthogonal keyboard;

• \( WPM_Q : WPM \) for a qwerty keyboard;

• \( \% \text{Gain} = 100 \frac{WPM_O}{WPM_Q} \): \( WPM \) percentual gain;

• \( KSPC_\alpha = \frac{\text{alphak}}{\text{alpha}} \): alphanumeric keystrokes per character, i.e., the mean number of alphanumeric keys needed to input an alphanumeric character;

• \( KSPC_\gamma = \frac{\text{alphak} + \text{other}}{\text{char}} \): keystrokes per character, i.e., the mean number of keys needed to input a whichever character, including a non-alphanumeric one;

• \( KSR_\alpha = 100(1 - KSPC_\alpha) \): alphanumeric keystroke saving rate, i.e., the mean percentage of alphanumeric keys saved;

• \( KSR_\gamma = 100(1 - KSPC_\alpha) \): keystroke saving rate i.e., the mean percentage of keys saved.

The \( KSPC_\alpha \) and \( KSR_\alpha \) parameters give a measure of the improvement with respect to alphanumeric characters only. Instead, the \( KSPC_\gamma \) and \( KSR_\gamma \) parameters take into account all the keys actually typed. \( KSPC_\alpha \) is the most significant parameter because it better shows the degree of optimisation that is due to the orthogonal keyboard. It is worth noting that the improvement is even underestimated, taking into account that the alphanumeric characters set contains digits and non-native letters, which do not belong to the p-syllable set. In the formulas for \( KSPC \) and \( KSR \), we have not included \( sh \) and \( shsel \), which are related to the shift keys for the capital letters.
and the $FR, CF, RF$ layouts, respectively. The term $sh$ has been excluded because the additional gesture for selecting a capital symbol is anyway required, independently on the kind of keyboard adopted, standard or orthogonal. On the other hand, the term $shsel$ has not been included because, with a proper implementation, shift keys can be selected concurrently with the task of inputting p-syllables or letters keys. In fact:

- in a hardware keyboard, the shift keys can be concentrated in a given zone, where the hand that is not used for typing can concurrently select the required shift;
- in a software keyboard, the shift keys can be allocated to the keyboard of the computer, where the hand that is not used for pointing can concurrently select the required shift.

Moreover, it is certainly true that the concurrent use of keystrokes requires a cognitive load. However, the basic reason for adopting KSPC is to provide a raw measure of the number of sequential gestures needed in typing. Therefore, the selection of a switch key, which is actually typed concurrently with another key, has not be taken into account because only a single temporal frame is in any case involved. We can note that the same consideration is valid for chord keyboards, where the concurrent selection of more keys must be done in the same temporal frame. In any case, the cognitive load that is due to the use of switch selectors is included in the time statistics that can be measured during end-user experiments.

The time parameters ($TO, TQ, WPMO, WPMQ, \%Gain$) have been used to measure and compare the typing times spent in real end-user experiments. In fact, although very important and significant, $KSPC$ and $KSR$ do not give a complete performance metrics, for they do not show the effect that the proposed methodology has on movement time and visual search time. Instead, this is possible by comparing the real typing times that are obtained by users typing the same text with different keyboards.

### 4.2 Experimental results

In the first experiment, the Italian and Spanish orthogonal keyboards have been tested with the automatic acquisition of a set of text files, which had been randomly chosen and downloaded from Internet (April 2004). It-Keyb has been tested with four texts taken from the electronic versions of *Corriere della Sera* and *Repubblica* newspapers. Sp-Keyb has been tested with four texts taken from the electronic versions of *ABC* and *El Día* newspapers. The results are outlined in Tables 5 and 6 (Curatelli and Martinengo, 2004).
In the second experiment, we have measured (in terms of $WPM$) the time improvement that is due to the orthogonal keyboard arrangement. For each reference language (Italian and Spanish)
Table 8 - Time statistics for qwerty vs Sp-Keyb

<table>
<thead>
<tr>
<th>User</th>
<th>$T_Q$</th>
<th>$T_O$</th>
<th>$WPM_Q$</th>
<th>$WPM_O$</th>
<th>%Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 1</td>
<td>3317</td>
<td>2322</td>
<td>8.14</td>
<td>11.63</td>
<td>42.84</td>
</tr>
<tr>
<td>User 2</td>
<td>4128</td>
<td>2878</td>
<td>6.54</td>
<td>9.38</td>
<td>43.42</td>
</tr>
<tr>
<td>User 3</td>
<td>3422</td>
<td>2217</td>
<td>7.89</td>
<td>12.18</td>
<td>54.37</td>
</tr>
<tr>
<td>User 4</td>
<td>3760</td>
<td>2516</td>
<td>7.18</td>
<td>10.73</td>
<td>49.47</td>
</tr>
</tbody>
</table>

we have compared the times that real non-expert users spent in typing the same text with both the standard (qwerty) keyboard and the orthogonal keyboard. The orthogonal Italian and Spanish keyboards have been implemented by using a modular programmable keyboard, with x-y matrix layout and long travel keyswitches. To obtain correct statistics the selected users should be really non-expert typers and the typing task should not significantly improve their expertise in typing. This has been met by selecting people with very low expertise in keyboard interaction, mainly limited to infrequent e-mail access, and by limiting the number of words users have to type during the test.

For Italian, we have selected four Italian-speaking people, aged 30 to 50. They have used both the standard qwerty and the orthogonal Italian keyboard to type the same short text corriere1-red, which is a strict subset of one of the texts already used in the first experiment (corriere1). This text contains 482 alphanumeric words (excluding separators and punctuation marks) and 3058 alphanumeric and non-alphanumeric characters. The results are outlined in Table 7, where, as already stated, $T_Q$ and $T_O$ are the total time in seconds for the qwerty and the orthogonal keyboards, $WPM_Q$ and $WPM_O$ are the WPM values for the qwerty and the orthogonal keyboards, while %Gain is the percentual gain in terms of WPM. For Spanish, we have selected four Spanish-speaking people, aged 30 to 50. They have used both the standard qwerty and the orthogonal Spanish keyboard to type the same short text abc1-red, which is a strict subset of one of the texts already used in the first experiment (abc1). This text contains 450 alphanumeric words (excluding separators and punctuation marks) and 2605 alphanumeric and non-alphanumeric characters. The obtained results are outlined in Table 8 (Curatelli et al., 2005).

5 Discussion

For both Italian and Spanish, Tables 5 and 6 show a significant improvement in typing alphanumeric characters. In particular, the $KSR_{\alpha}$ values stay between 44.32% and 45.20%, for the Italian texts, and between 43.37% and 43.93%, for the Spanish texts. The slightly better gain obtained
with the Italian texts is probably due to the higher frequencies of double consonant graphemes in Italian. Concerning $KSR_\gamma$, Tables 5 and 6 show anyway significant gains, ranging between 35.71% and 36.39% for the Italian texts, and between 34.22% and 35.65% for the Spanish texts. Although less representative of the gain achieved in typing alphanumeric characters, the $\gamma$ parameters show the degree of global improvement as it is perceived by the user. For highly inflected languages, such as Italian and Spanish, these gains are comparable with those that can be obtained by using state of the art prediction tools when the visualisation of one prediction is selected.\footnote{We have considered single prediction to show that the orthogonal keyboard method is as efficient as the predictive methods not involving the cognitive selection among different choices.}

For example, the FASTY prediction system yields some 35% $KSR$ for inflected languages, such as Dutch, French, German and Swedish. In fact, the $KSR$ values reach 50% only when 5 or 7 predictions are visualised. (Zagler et al., 2003). Higher gains can be obtained in English; but this is due to the fact that it is an almost uninflected language.

Concerning the second experiment, the measured time improvements are given by the $\%Gain$ values in Tables 7 and 8. They show significant improvements, ranging between 44.58% and 52.20% for the Italian text, and between 42.84% and 54.37% for the Spanish text. Moreover, for all the users involved in the test, these values are significantly better than the $KSR_\gamma$ values obtained, in the first experiment, with the corresponding full texts. This additional gain (in any case more than 25%) is very important because it does not depend on the gain due to the use of p-syllables, which is already outlined by $KSR_\gamma$. This result shows that the orthogonal keyboard paradigm allows non-expert users to obtain faster visual search times than those obtained with standard qwerty keyboards. In fact, the proposed orthogonal scheme allows the non-expert user to reach the correct $x,y$ positions of a key, without the need of a complete 2-D visual search of the keyboard and using his natural expertise in lexicographic ordering. Instead, the standard qwerty arrangement requires a quite higher time overhead to locate the keys, which are placed randomly on the keyboard layout. It is worth noting that all the people involved in the tests had not done any previous typing training on orthogonal keyboards while they had already infrequently used qwerty keyboards. This shows that the orthogonal framework gives the non-expert user a much higher fluency in typing.

The above results are very encouraging because they show that a significant gain in text typing can be obtained at the keyboard level, by adopting a keyboard framework that is well suited for memorisation and fast use by non-expert users. Concerning motor-impaired users, the experimental validation of the method requires to perform specific tests, which are by far more difficult to
plan and make. However, as the orthogonal scheme affects typing in the same way for both non-expert and motor-impaired users, it is arguably that also motor-impaired users take benefit from the proposed scheme.

It is worth noting that additional improvements can be obtained through word completion and prediction techniques, space adding tools, and intelligent phrase understanding methods. For example, let us suppose that the orthogonal keyboard framework produces a gain given by $K_{SC}\gamma = 0.6$ and $K_{SR}\gamma = 40\%$. Moreover, let us suppose that the word prediction tool produces, when typing with a standard keyboard, a gain given by $K_{SC}\pi = 0.6$ and $K_{SR}\pi = 40\%$. Therefore, the use of both the orthogonal keyboard and the predictive tool would yield an overall improvement given by $K_{SC}\tau = K_{SC}\gamma \times K_{SC}\pi = 0.36$ and $K_{SR}\tau = K_{SR}\gamma \times K_{SR}\pi = 64\%$. However, it is arguably that the $K_{SC}\tau$ and $K_{SR}\tau$ values would be even better, provided that the predictive tool is specifically designed for p-syllabic text entry units. In fact, if words are built with a lower number of text entry units, the predictive tool can better explore the (reduced) space of solutions.

A possible objection to our approach is that the practical usefulness of a new text-entry method is affected by its localizability, as computer devices and tools are more and more global. However, in our case this criticism is true only when applied to the implementation of the orthogonal method for a specific target language. Instead, the orthogonal text entry paradigm is in itself a general method, because it can be directly applied to the graphemic structure of whichever alphabetical language with transparent orthography. This means that for most alphabetical languages in the world we can easily implement orthogonal keyboards that are tailored to the users’ own native language. Therefore, the suitability of the method for a very large number of different native languages will greatly increase its usefulness in a world in which more and more people have access to computers.

Another issue concerns the possible applicability of the orthogonal approach to non-transparent languages, in particular to English, which is one of the most opaques languages in the world. As already stated, language transparency implies an almost one-to-one correspondence between a phoneme and a grapheme. This, in turn, makes it possible to build a deterministic keyboard layout, i.e., a keyboard in which the user can, immediately and without ambiguity, map a given pseudo-syllable sound to one key only. This is not the case of English, in which a specific phonemic couple /C//V/ can be associated to more graphemic couples $\hat{C}\hat{V}$ and, conversely, a specific graphemic couple $\hat{C}\hat{V}$ can be associated to more phonemic couples /C//V/. Therefore, the definition of an adequate p-syllable set would require the insertion of a very large number of p-syllables, so lead-
ing to huge keyboard sizes. Moreover, English has much more syllables and syllable structures than other languages (see Section 2.1). This means that a lot of frequently occurring syllables do not belong to the p-syllables paradigm and should be included in the keyboard to obtain an adequate coverage. As a consequence, although very effective for transparent languages, the proposed approach does not seem to be well suited for English. The same considerations can be done to evaluate the applicability of the orthogonal approach to other non-transparent languages.

6 Conclusions

This paper has described a powerful pseudo-syllabic paradigm for improving typing efficiency in languages with transparent orthographies. We have shown how, through the use of a novel orthogonal keyboard framework, non-expert users can input pseudo-syllables quickly and intuitively, by independently processing the vowel and consonant graphemes of each pseudo-syllable. Two transparent languages, Italian and Spanish, have been considered and two orthogonal keyboards have been designed for the two languages. The keyboards have then been implemented both as software and hardware keyboards and tested with a simulation experiment and an end-user experiment. For both languages, a nearly double improvement in typing alphanumeric characters has been found, comparable with the improvements that can be obtained in inflected languages by using word prediction tools with a single prediction selected. The two techniques can be used together to reach global $KSR$ values exceeding 60%, without the need of performing any visual exploration in the prediction window. Even better improvements have been obtained in the second experiment, in which we have compared the total typing times that non-expert users have spent with orthogonal keyboards and standard qwerty keyboards.

In conclusion, the improvements that can be obtained for transparent languages are very encouraging. Future research will concern the implementation of the paradigm with optimised software and hardware keyboards, the application of the orthogonal keyboard framework to other languages, and the extension of the p-syllable model to better characterise the syllable structures of transparent languages.

Acknowledgements

The authors wish to thank Prof. Mayora-Ibarra from ITESM, Cuernavaca, Mexico, for the contribution on testing the Spanish keyboard, and the useful comments and suggestions of the referees.
References


url=http://www.phon.ucl.ac.uk/home/sampa/home.htm


