Text Input Disambiguation Supported on a Hierarchical User Model

Carlos Bento and Nuno Gil

Centro de Informática e Sistemas da Universidade de Coimbra
Polo II da Univ. de Coimbra
bento@dei.uc.pt – nuno@gilito.com

Abstract

Mobile phones are used for various tasks that go far from voice communication. A popular use is for composition of short messages (SMSs), but other applications are also available like email, agenda, contact, and note management. All these uses have in common the need for text input on a small keyboard with ambiguity problems.

Various techniques are currently used for input disambiguation, with variable results in terms of usability and efficiency. Some techniques achieve good performance with messages composed by words from a dictionary but poorly when a significant number of words are not in memory.

In this paper we present a solution for text disambiguation supported on a general model for disambiguation plus a user model generated from previous messages sent by the user. We present results obtained with this approach and discuss future improvements.

1. Introduction

The use of Short Message Services (SMS) is an important source of revenue for the mobile communication industry.

With the growth in the processing capacity of mobile terminals, several new applications have emerged. For many of these applications the success depends on the availability of efficient ways for text input in 12-key phone keyboards. This is especially true for applications that involve composition of long sequences of text like when preparing emails or writing notes.

In general, current devices incorporate one or both of two groups of text input techniques [1]: pen-based and key-based.

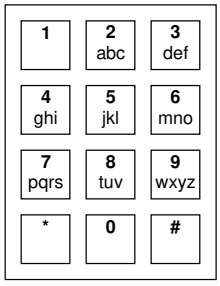

Figure 1: Twelve keys ambiguous keyboard.

The first one, more common in Personal Data Assistants (PDAs), makes use of one "pen" and a sensitive screen allowing inputting text in two different ways: on a virtual keyboard that shows up on the terminal screen or drawing the symbol that must be recognized as the letter we want to write.

The key-based approach, more common in mobile phones, can be implemented on complete keyboards, i.e. keyboards with one key for each letter (e.g. Nokia 9110 Communicator) and ambiguous keyboards, namely 12-key phone keyboards, in which each key is assigned to three or four different letters (e.g. Nokia 7110).

The focus of our research is on the second group of input systems: ambiguous keyboards. Fig. 1 shows the usual aspect for these keyboards.

The way letters are disposed is defined by international standards[1], so independently of the language of use all users have the same ambiguities on their keyboards.

The typing process in such devices involves pressing each key one or more times until the desired letter is retrieved. The number of times one key has to be pressed depends on the sequence of letters predefined for this key. For instance, to write the letter *y* the *9* key has to be pressed three times. This simplified way of entering text is called multi-tap [2], since we need to *tap* several times each key to obtain the desired letter.

Fig. 2 represents the sequence of keys that need to be pressed to write the word "terminal" using the multi-tap technique.

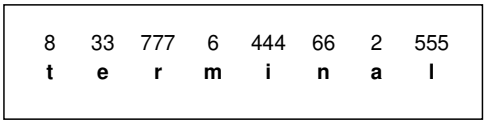

Figure 2: Sequence of pressed keys to write *terminal* with *multi-tap.*

Using the multi-tap technique we would need to press the keys 16 times to write this 8 letters word. Luckily, there are several *intelligent* techniques that allow writing text with a much smaller number of key strokes.

The most known technique is T9[2] [3]. This technique is based on a dictionary. Working with T9 is quite simple, although some training is necessary. The user just needs to press one time each key associated to the letter that s/he wants to write. The system progressively tries to find the words (or part of the words) in the dictionary that include letters associated to the keys that were pressed so far.

If all the keys were pressed and the word retrieved from the dictionary is not the one that we want, a special key must be pressed to retrieve the other possible words, and hopefully, the desired word will be presented.

To write words that are not in the dictionary, it is necessary to turn off the *intelligent writing* and write as described for the multi-tap technique. In recent implementations of T9, the system integrates the new words in the dictionary.

In Fig. 3 we represent the writing sequence of the word "terminal" using T9 with an English Language dictionary.

[1] ITU E.161 ou ANSI T1.703-1995/1999 ou ISO/IEC 9995-8:1994.
[2] Acronym for *text on nine keys*.

key	result
8	t
3	ve
7	ter
6	term
7	termi
6	vermin
2	termina
5	terminal

Figure 3: Sequence of pressed keys to write *terminal* with T9.

In this picture it is patent that only 8 key-presses were necessary to write this 8 letter word.

Being T9, the technique with the strongest market share, there are other techniques competing with it namely eZiText [4], iTap [5] and LetterWise [6].

There are also other approaches that are not so well known. Some of them are based on software key disambiguation like Less-Tap [7] and HMS [8], and techniques in which the disambiguation is performed by the use of extra keys like chording keyboards [9], or by the use of inertial sensors like TiltText [10].

Systems based on a dictionary such as T9 have important limitations [11] like the lack of several common words in the dictionary; no slang words in the dictionary; the need to visually follow writing, the need to switch between language databases, and the need for teaching new words *manually*.

A well known disambiguation technique that is not based on a dictionary is LetterWise [12]. Instead of a dictionary it calculates probabilities that are used to guess the next letter to be written having in consideration what was written so far (the prefix).

For instance, if we want to write the word *the* and we have so far written *th* and key *3* is pressed, based on the probabilities for the English Language, the returned letter will be *e* instead of *d*. If the returned letter is not the desired one there is the need to type the special *NEXT* key so that the next letter with the highest probability can be retrieved.

One problem with LetterWise is that as soon as the probability matrix for the various prefixes is built from a language corpus it becomes difficult to write non-words like abbreviations or words not present in the initial corpus. Another limitation, related with the first one is that LetterWise does not have sophisticated learning capabilities. If a user inserts non-words or words that are not covered by the probability matrix the system does not change the probability matrix in order to recognize these words. This results into the burden of pressing the *NEXT* special key many times.

The emergence of more powerful mobile terminals makes possible to consider more elaborated disambiguation techniques. In our approach we focus on the integration of a learning mechanism in non dictionary based systems like LetterWise. Additionally we assume that a user writes different types of messages dependent of the interlocutor. We call these types of messages by message styles. For instance it is supposed that a kid communicating with other kids of the same age uses a lot of abbreviations and slang but when communicating, for instance, with an adult person tends to use a quite different lexicon.

With this in mind we propose an approach for text input disambiguation supported on a user model. The user model comprises various profiles. Each profile has a representation similar to the probability matrix used by LetterWise. One main advantage of this approach is that the profiles are learned dynamically along the composition of new messages. Another is that the user model comprises various profiles organized in a hierarchical structure. This makes possible to have different disambiguation policies dependent on the type of message that is being typed. Messages from a more general style are disambiguated by the profile in root of the hierarchical structure and messages written in more specific styles are analyzed by the descendent profiles.

In section two we describe our approach. Section three presents the results that were achieved with a corpus of synthetic messages comprising two different styles of messages. In the third section we discuss the results that were obtained and outline future improvements. Section four summarizes the main conclusions from this work.

2. Disambiguation supported on a hierarchical user model

As it was already pointed out common users compose various types of messages going from well structured phrases, respecting language rules, to messages full of jargon and abbreviations. This makes appealing to support disambiguation on a model of well formed phrases plus a user model that represents the specific styles used by this person.

These requirements motivate us to consider a model comprising various profiles structured in a hierarchical way going from a general profile modelling a corpus of general messages to more specific profiles associated to each type of messages produced by a specific user.

2.1. User profiles

Within our approach a user profile corresponds to the probability matrix used in LetterWise. In Fig. 4 we show the messages that were in the origin of the profile represented in Fig. 5.

```
cannot contact you call me back
in lesson call you back later
call me when you are free please
oh my god i hope this finishes soon
```

Figure 4: Four messages.

Each line in this matrix represents the number of occurrences of the various letters of the alphabet conditioned by a specific prefix. For instance the numbers 1, 1, and 1 in the third row of Fig. 5 represent that within the corpus of messages used to learn this profile the letters *a*, *b*, and *c* occurred one time, after the occurrence of prefix *u* and of pressing key #2.

For text disambiguation each time we have a prefix and key pressed we follow the respective key column and prefix row and return the triple or quadruple with the probabilities for the letters associated to this key. The disambiguation occurs by selecting the letter for his key with the highest number of occurrences.

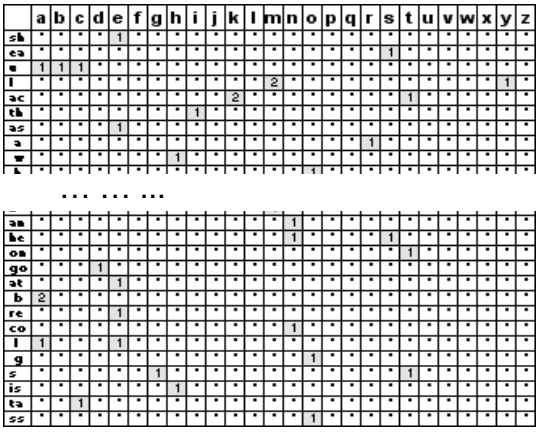

Figure 5: A profile generated from four messages.

In the tests presented in this paper we consider profiles produced considering prefixes of one and two letters length.

2.2. Profile hierarchy

Fig. 6 shows a user model comprising a set of profiles organized in a hierarchical way.

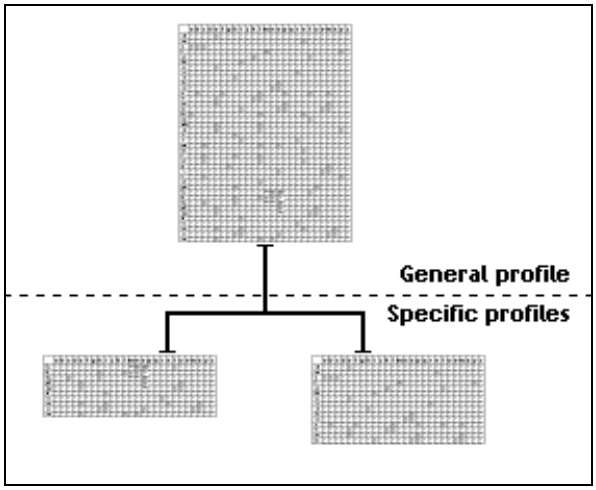

Figure 6: A hierarchy of profiles.

At the beginning (before the user types any message) the profile in the root of the hierarchy represents the occurrences of letters after a prefix for all the messages in the training corpus.

The specific profiles only deal with user messages. They represent the occurrences for the groups of messages sharing common characteristics but apart from each other – different styles. In this way this hierarchy corresponds to a process of hierarchical clustering. The criteria for clustering can assume various forms. One could be to create two specific clusters for messages with average word length bellow 3 letters and another for messages with average word length equal or greater to five[1]. In this case messages with an average length between 3 and 5 would only be considered in the general profile at the root of the hierarchy. The root profile is influenced by the messages in the specific profiles plus the messages that were not considered in these profiles.

In the following subsection we describe how the global and specific profiles are produced from a corpus of messages and user messages.

2.3. Putting all together

In this subsection we start describing how a general profile is created. Then we present how a new message updates profiles and occasionally creates new specific profiles. We finish with the description of the disambiguation of a message using the profile hierarchy.

In our approach the first version of the general profile is created from a corpus of messages that covers the kind of SMSs produced by common users. Each message is scanned for the number of occurrences of each letter after the prefixes of length one and two that precede this letter in the message. Anytime a prefix does not exist yet in the profile a new row is created and the prefix is added to the profile.

Only the general profile is present when the user starts using the device. For the first *n* letters of the user message[2] the general profile is used for disambiguation. This means that after the second letter of the message the system looks for a prefix of length two comprising the previous two letters inserted in the message plus the key that is being pressed and tries to disambiguate from the occurrences assigned for the candidate letters (the letters associated to the pressed key). This means that the letter associated to the pressed key that has the highest number of occurrences in the general profile is the one selected for disambiguation. If this guess is correct than the number of occurrences associated to this letter in the profile is increased by one. If this is not the correct letter then the user has to tap again for retrieval of the next letter with the highest number of occurrences in the profile till the desired letter is retrieved.

After the insertion of the n^{th} letter for the message the system has to decide which profile will be used for disambiguation of the coming key strokes. This decision is based on the similarity of a profile created with the *n* letters insert till the moment against the profiles in the profile hierarchy.

The profile for the *n* letters of the message is created the same way as the other profiles. This means that we scan the *n* letters and count the occurrences of the letters conditioned by their prefixes of length two and one. Fig. 7 shows the profile for the first 15 letters of the first message in Fig. 4:

cannot contact you call me back

The similarity metric used for decision on which profile will be used to disambiguate the letters after n^{th} letter is based on the average of ranking coefficient correlation for the triplets or quadruplets addressed by the same prefix/key in the message profile and each hierarchical profile.

[1] We experimentally concluded that this is not a good criterion.

[2] Currently *n* is assigned to 15 letters.

	key #2			key #3			key #4			key #5			key #6			key #7				key #8			key #9			
	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
c	1														1					1						
a			1											1												
ca														1												
an														1												
n														2	1					1						
nn															1											
no																				1						
o																				1						
t_			1																							
_			1																							
_c															1											
co														1												
on																				1						
nt	1																									
t	1																									
ta			1																							
ac																				1						

Figure 7: Profile for the message "cannot contact …".

Only triplets and quadruplets that have at least one occurrence for one letter associated to the respective key in the message profile and hierarchical profile contribute to the average value.

For calculation of the ranking similarity we used the Spearman's Ranking Correlation Coefficient [13] which gives values between 1 and -1:

$$\text{Spearman Coef} = 1 - \frac{6\sum_{i=1}^{n} d^2}{n^3 - n} \quad (1)$$

where n is *3* and *4*, respectively, for triplets and quadruplets, and d is the distance in terms of ranking of a letter in the triplets/quadruplets under evaluation. For instance, for the triplets in the first row of Fig. 8 the distance for letter *a* is *2*, for letter *b* is *2* and for letter *c* is *1*.

In Fig. 8 we represent two extreme values returned by the Spearman's Coefficient. In the first case we have two triplets for key #2, respectively, in the message profile and in a hierarchical profile. The ranking of the number of occurrences for letters *a*, *b*, and *c* is opposite in the message and hierarchical profiles so the ranking correlation coefficient is -1.

The other situation in Fig. 8 represents a ranking of the number of occurrences that is similar in both profiles so the ranking correlation coefficient is 1.

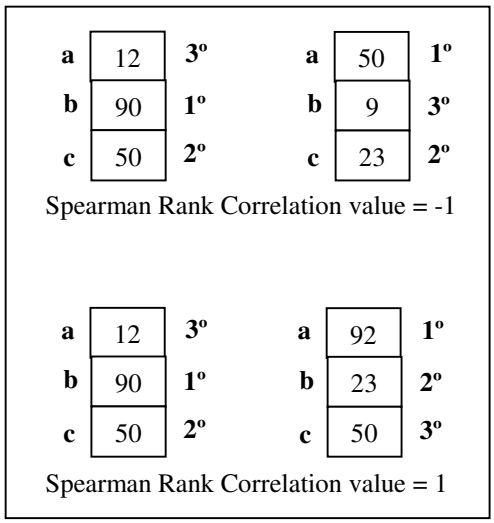

Figure 8: Rank matching between corresponding triplets in two profiles.

As pointed before, the similarity value between two profiles is calculated by summing up the Sperman's Coefficient for the triplets or quadruplets with at least one occurrence value different from zero in both profiles[1], divided by the number of triplets and quadruplets that are considered.

The hierarchical profile with the highest similarity against the message profile is the one selected for disambiguation of the remaining part of the message. An exceptional situation occurs when the selected profile is a specific one and the prefix under consideration is not present in this profile, but is present in the global profile. In this case the global one is used for disambiguation for this letter. The disambiguation process returns to the use of the specific profile for the following letters.

When the user finishes composing the message, the message profile is updated with the part of the message after the n^{th} letter. The next step is the recalculation the similarity between the message profile and all profiles in the hierarchy. If none of the similarity values concerning all the existing profiles is above a certain threshold this means this message profile is quite different from all the profiles present in the user model. This determines that the message profile is added to the hierarchy of profiles.

3. Experimental results

The main assumption for this approach is that users adopt different styles for messages according to groups of interlocutors. With this in mind we started by creating two sets of synthetic messages. Bellow we present the synthetic corpus of messages that we used and the results that were obtained. In the following section we discuss these results and on how we think they relate to real user's messages.

In the following tests we were mainly interested in understanding the kind of messages that benefit from using a hierarchy of profiles vs. only a global profile.

Our main intuition is that the use of a single global profile has as a consequence that letters that fewer times come after a certain prefix will not be correctly disambiguated. If we have specific profiles in addition to the global profile these minority messages have the possibility to be correctly disambiguated. The tests we performed till now corroborate this assumption.

3.1. Corpus of synthetic SMSs

For this experiment we considered a corpus of synthetic SMSs comprising two groups of messages with the following structure:

GROUP #1:
$<X_1><X_2>< L_1><X_1><X_2>< L_1>\ldots <X_1><X_2>< L_1>$

GROUP #2:
$<X_1><X_2>< L_2><X_1><X_2>< L_2>\ldots <X_1><X_2>< L_2>$

With X_1 and X_2 letters from the alphabet and L_1 L_2 different letters but associated to the same key[2].

We produced about two times the number of messages from GROUP #1 than from GROUP #2 in order to create a major group (GROUP #1) and a minority group (GROUP #2).

[1] This number different from zero can occur in different letters in the two profiles.

[2] This is the condition to produce an ambiguity.

3.2. Results

We run the experiment on the synthetic corpus of messages with three variants. First we only considered profiles with prefixes of length one (we hidden the prefixes of length two). The second test comprised only prefixes of length two. The third test comprised both types of prefixes. For these tests we measured the KSPC (Key Stokes Per Character) value, which represents the average number of key strokes per message character. The obtained results are described in Fig. 9.

```
Prefix Length:1     GLOB PROFILE / GLOB+SPEC PROFILES

  ABY                  1.0000000 / 1.0000000
  ABX                  1.3333334 / 1.1041666
  ABY + ABX (50%/50%)  1.1666666 / 1.0520834
  ABY + ABX (33%/77%)  1.2222222 / 1.0694444

Prefix Length:2     GLOB PROFILE / GLOB+SPEC PROFILES

  ABY                  1.0208334 / 1.0208334
  ABX                  1.3541666 / 1.1250000
  ABY + ABX (50%/50%)  1.1875000 / 1.0729167
  ABY + ABX (33%/77%)  1.2430555 / 1.0902770

Prefix Length:1+2   GLOB PROFILE / GLOB+SPEC PROFILES

  ABY                  1.0000000 / 1.0000000
  ABX                  1.3333334 / 1.1041666
  ABY + ABX (50%/50%)  1.1666666 / 1.0520834
  ABY + ABX (33%/77%)  1.2222222 / 1.0694444
```

Figure 9: Results.

In the following section we comment on the above results.

4. Discussion

We start analysing the results obtained considering only the major group of messages (GROUP #1) for tests. This group comprises messages with thirty letters comprising a sequence of *ABs* followed by *Y*. These messages are correctly disambiguated by the general profile as they are dominant. So if we use profiles of length one they are always disambiguated (KSPC = 1). When we use only prefixes of length two they are not correctly disambiguated only for the first B in the message and this is the reason why we do not have precisely 1 for result (KSPC = 1.02). When profiles of length one and two are considered then we have the first situation (KSPC = 1).

Now let see what happens with the minority messages alone. When using prefixes of length one, two, or both and only the global profile is used the system completely fails disambiguating the *X* letter (KSPC = 1.33). It means that *A* and *B* are disambiguated but *X* is never correctly suggested (this makes the increase of 1/3 = 0.33 in the KSPC value). The value 1.35 for the KSPC when only prefixes of length two are used is due to the first *B* in the message that is not correctly disambiguated.

When we mix messages from both groups the results obtained are between the extremes explained above. This is what is expected. We have a degradation of the performance when only the global profile is used and an increased number of minority messages are typed.

These results make evident the need for specific profiles for messages which pattern is minority against the patterns represented in the general profile.

From these results we conclude that in situations in which a major style of messages coexists with other styles that comprise distinct patterns of letters we need to consider a hierarchical system of profiles (probability matrixes) for the disambiguation model in order to have the minority messages correctly disambiguated.

At the current stage of this work we did not have yet the opportunity to use real corpus of messages integrating different styles. Although we think that starting by producing corpus of synthetic messages is a good way to understand the advantages and limitations of this approach.

Another aspect that needs further study concerns the criterion for creation of new specific profiles during the learning step. From the experiments we performed till now it becomes evident that simply using as criteria a threshold for the similarity value does not result into good specific profiles. We think that the extensive work that has been performed in the area of hierarchical clustering can be inspirational for the improvement of this process.

5. Conclusions

Although various methods are currently used for text disambiguation in 12 key phone keyboards and other ubiquitous devices many of these techniques are dictionary-based which result into poor performance when jargon or abbreviations are extensively used in a message.

An approach that is not dictionary-based is LetterWise [12]. It performs satisfactorily on balanced corpus of messages. Notwithstanding when the probability matrix is trained with a majority of messages in a certain style the disambiguation completely fails for minority messages in a different style. By a different style we mean one that for various sequences of prefixes has a diverse disambiguation letter from the ones in the other styles.

In this paper we propose a hierarchical profiling (hierarchy of probability matrixes) for disambiguation used to disambiguate messages written in minority styles.

We test this approach with a corpus of synthetic messages comprising a major and a minority style. We show that single profile approaches like LetterWise perform poorly on the minority messages and that with an hierarchical structure of profiles this problem can be overcame.

Our approach has two main strengths. One is that it maintains a model of the various styles of messages composed by the user and absent in the initial corpus of messages used to initialise the system. This is of great relevance especially when the user integrates a great amount of slang in the messages which is a situation very common in SMS communication. Another important aspect has to do with the learning capabilities associated to this approach.

The system learns in two ways. It learns a new profile when the message on its origin is quite different from the ones represented by the profiles in memory. In another way the system learns by updating the number of occurrence in the existing profiles along the composition of messages by the user. The possibility of creating new profiles has as a consequence that minority styles are not override by major styles. This effect is stressed in the experimental results presented in this paper.

Although we used the Spearman Coefficient as being a suitable approach for similarity ranking between two arrays of letter occurrences, the work we are now developing shows that this is a very computationally heavy approach in concern of CPU usage. Being so, a new lighter approach is being developed.

In the future we want to test this approach with real messages with the characteristics described in this paper. Another aspect that needs attention concerns to in which circumstances we must create new specific profiles.

Also we want to test the impact of having few specific profiles versus increasing the number of profiles. Another aspect concerns the length of the prefixes that integrate the profiles. Our experience says that there is not a great improvement when we go from a length limit of three to four, with the handicap of a dramatic increase in memory occupation.

Also it is important to understand if it is acceptable to discard prefixes of length one which have low discriminatory power and substantially increase the computational complexity.

Another thing that couldn't yet be done, but that has high relevance, is the performance comparison with the existing approaches.

References

[1] Poika Isokoski (2004). *Manual Text Input: Experiments, Models, and Systems*. Academic Dissertation, Faculty of Information Sciences of the University of Tampere, April 23rd, 2004. A-2004-3 Tampere. ISBN 951-44-5959-8.

[2] J. Cardinal and S. Langerman (2005). Designing small keyboards is hard. *Theoretical Computer Science*, 332:405-415.

[3] T9: Learn more. Tegic Communications, Inc, 30 April 2005 http://www.t9.com/learn.html.

[4] eZiText word prediction. Zi Corporation, 30 April 2005 http://www.zicorp.com/ezitext.htm.

[5] Whaley D. (2002). *Better Ways of Typing Text Messages on your Hand-held Device, in Business Briefing: Wireless Technology 2003,*p. 1-3

[6] MacKenzie, S., Kober, H., Smith, D., Jones, T., Skepner, E. (2001). *LetterWise: prefix-based disambiguation for mobile text input*, in Proceedings of the 14th annual ACM symposium on User interface software and technology, Orlando, ACM, p. 111-120.

[7] Pavlovych, A., Stuerzlinger, W. (2003). *Less-Tap: A fast and easy-to-learn text input technique for phones,* in Proceedings of Graphics Interface, in GI 2003, Canadian Information Processing Society, p. 97-104.

[8] Hasselgren, J., Montnemery, E., Nugues, P., Svensson, M. (2003). *HMS: A Predictive Text Entry Method Using Bigrams* , in Proceedings of the Workshop on Language Modeling for Text Entry Methods, 10th Conference of the European Chapter of the Association of Computational Linguistics, Budapest, p. 43-49.

[9] Wigdor, D., Balakrishnan, R. (2004). *A Comparison of Consecutive and Concurrent Input Text Entry Techniques for Mobile Phones,* in Proceedings of CHI 2004, Volume 6, Number 1, Vienna, ACM, p. 81-88.

[10] Wigdor, D., Balakrishnan, R. (2003). *TiltText: using tilt for text input to mobile phones,* in Proceedings of the 16th annual ACM symposium on User interface software and technology, Vancouver, ACM, p. 81-90.

[11] Gutowitz, H. (2003). *Barriers to Adoption of Dictionary- Based Text-Entry Methods: A Field Study,* in 10th Conference of the European Chapter of the Association for Computational Linguistics, Budapest.

[12] MacKenzie, S. (2002). *KSPC (Keystrokes per Character) as a Characteristic of Text Entry Techniques,* in Proceedings of the 4th International Symposium on Mobile Human-Computer Interaction, Springer-Verlag, p. 195 – 210.

[13] On-line statistics, University of Leicester. 30 April 2005 http://www.le.ac.uk/biology/gat/virtualfc/Stats/spear.htm