Intelligent Tamil texting for Mobile environment

Baskaran Sankaran [1], Srinivasan K [2], Ramesh Kumar S [3]

[1]AU-KBC Research Centre, Chennai, India; [2]Infosys Technologies Limited, Chennai &
[3]Electronic Data Systems, Chennai, India

[1] baskaran@au-kbc.org, [2] Srini_Krishnamurthy@infosys.com,
[3] rsurulivelu-eds@eds.com

Keywords

Mobile computing, Text input, Predictive text input, word prediction, word completion, Tamil keypad

Abstract

The limited scope of the input interface in mobile computing environment has led to the development of intelligent input mechanisms. These mechanisms can be classified into two categories - text input and graphic input. While graphic input methods are widely used in PDAs and palm pilots, text based interfaces are more appropriate for mobile communication devices. Tap-a-tap, multi-tap, predictive text input etc. are some of the popular text input mechanisms used in mobile computing devices such as simputers, PDAs and mobile phones. This paper broadly focuses on the text-based input solutions for the mobile communication devices.

Messaging or more simply *texting*, is one of the key features in mobiles and the limitations of the telephone keypad pose a greater challenge in keying in texts. The common solution is multi-tap approach, where the characters are spread across the keypad and the user has to press the keys appropriate number of times for a specific character. The disadvantage with this approach is the slow typing, waiting time etc. Keying in a substantially long message will often result in fatigue. Several research works focused on finding a better and efficient alternative and this resulted in the predictive text input algorithms. The idea here is to use the optimized language model for intelligent texting, which improves the input speed tremendously. Predictive text input has been already developed for about 40 languages including English, Hindi and Tamil though the commercial version for Tamil is yet to be made available in any device.

First, we point out the challenges in keypad design and text input methods in general and also specifically for Tamil. We then develop and demonstrate the predictive text input algorithm. We also propose to improve the predictive texting algorithm for Tamil by adding several new features such as auto completion, next word guessing etc. This paper also presents a schematic on Tamil mobile keypad taking into consideration input speed, accommodating all characters, ease of use etc. This work has practical relevance for the mobile industry and will improve the Tamil texting approach by leaps and bounds. Our schema automatically learns new words that are initially not available in the language model from the user. The algorithm will be implemented on a regular desktop computer and we hope that this can be suitably optimized for the mobile environment easily.

1. Introduction

Text input in mobile environment has always been challenging since the days of PDAs. The challenge arises because of limited space available in handhelds, which prohibits the option of having a full-fledged keyboard as in computer. The current phenomenon rapidly moves towards the integration of wireless telephony & handheld computers and this has necessitated intelligent and easy-to-use mechanisms for text input. These mechanisms form one of the key requirements in mobile applications including messaging, mobile commerce, web access through mobile etc. The solution proposed should also satisfy several criterion suitable for mobile environments such as compactness (in terms of physical and memory storage size), simple to use, adaptive etc.

Several mechanisms have already been developed for English and other major European languages. These include T9, predictive text input, extended keypad; graphic based text-input etc. Some of these mechanisms have already been integrated as features in mobile phones by various phone manufacturers for English, French, Spanish, German and host of other languages.

The rapid growth of the mobile phone industry in India underscores the need to develop such intelligent texting solutions specifically for Indian languages, especially because of lower penetration of English. Developing texting mechanism for Indian languages, similar to Chinese and other Asian scripts is more complex and challenging because of the larger character-set for these languages. Tamil, which is supposed to have the smallest character-set among the Indian languages, have at least 247 characters (excluding the Grandtha chars). This requires an intelligent design to locate the basic characters directly in the keypad and a mechanism to generate other characters by a combination of key presses. This paper discusses an intelligent texting mechanism for Tamil in mobile platform, especially phones, loaded with several language specific features for high-speed typing, while making it user-friendlier. Though we have designed and developed the framework for mobile devices, the implementation is done on the normal desktop computers. However, we are sure that this can be extended to mobile platform by suitably changing the software and implementation based on the same framework.

This paper is organised as follows. Section-2 details out the challenges involved in designing a keypad of mobile devices including those that are specific to Tamil. Section-3 explains the current start-of-the-art in the mobile texting, where we discuss the existing techniques and their applicability for Tamil. The penultimate section 4 is about the system implementation, covering some statistics and some additional features of the system. Concluding remarks are given in the final section.

2. Design Challenges

Any approach to text entry in mobile devices is based on at least one of these optimization techniques, viz. i) movement minimisation and ii) language prediction [6]. Movement minimisation models the targeted human movements to reduce the movements of the finger or pen interacting with a mobile device. The language prediction strives to reduce the input burden by predicting what the user intends to enter by exploiting the statistical nature of the language.

2.1 Keypad Design

The Fitts' law models the human psychomotor behaviour and it is based on time and distance [9]. It predicts the movement time based on the distance moved by the finger to reach a specific target (key) and the size of the target. It is mathematically expressed as,

$$MT = a + b \log2 (2A/W) \quad (1)$$

where, MT - *Movement Time*
a, b - *Regression coefficients*
A - *Distance of movement from start to target center*
W - *Width of the target*

Fitts' law is largely used in design of user-interfaces and is used to measure the performance of movements in monotonous tasks (as in assembly lines), with the aim of reducing the stressful movements. It should be noted that this applies for rapid and aimed movements (Eg. typing, picking a specific object from a specific place etc). It is interesting to note that, Fitts' law is also applicable in user-friendly web designing [10].

Though Fitts' law is very useful in user interface design and is employed in several handheld devices, it is generally not applied for the character arrangement problem in mobile phones for the following reasons.

- Limited no. of keys in mobile phones, requires that each key be assigned more than one character. So, if the characters are scrambled in order to design a simple keypad as the QWERTY, the users will have to learn the new methodology. History has one simple message - any new device that expects the users to put in efforts in learning to use it, is bound to fail.

- Mobile phones are largely meant for single hand usage and hence the introduction of touch-typing similar to QWERTY will not be use-friendly, especially since the users also support the phone using the same hand.

However, Fitts' is used in performance evaluation studies for different text input methods as has been used in Silfverberg [7]. Because of these reasons, the English characters are arranged alphabetically in the 12-key telephone keypad following the international standard [2, 3]. In this schema each character shares the same key with at least two more characters apart from the number. However, Chicagologic has released a keypad Delta-II [11] that retains the QWERTY design albeit with slight differences. Here, the characters are arranged in a 5 x 6 keypad with some alphabets sharing the keys with number keys. Information about any commercial deployment of Delta-II by mobile phone manufacturers or any publication with systematic evaluation is not known till now.

In this paper, we decided to retain the popular 12-key mobile keypad because of its wide acceptability and proven commercial success.

2.2 Language Modelling

Language modelling exploits the statistical nature of the language and constantly attempts to predict subsequent characters in a word or in some cases the next word itself based on the

already typed sequences. It is based on the frequent words in a language and also on the N-grams (bigrams, trigrams and so on) gathered by analysing a large collection of documents covering wide genre - corpus. However, there are few caveats that should be given attention in basing the language model on a standard corpus as in [6]. i) The corpus may not be the representative sample of the user language, ii) the corpus does not reflect editing process and iii) corpus does not capture input modalities.

We discuss the first point in the section given below and we skip the other two since we believe that they are not too significant and readers are encouraged to refer MacKenzie [6] for full details.

2.2.1 Corpus - Not a Representative Sample of User Language

"Corpus - Not a Representative Sample of User Language" - This stems from two factors, i) the language model doesn't include the spaces and punctuations and ii) the characteristics of the text entered by the user is highly conditioned by the nature of application. The first factor stresses the point that most of the language models fails to capture the language to a better degree of perfectness, since they do not account the spaces and the punctuation symbols in a text. Though this is generally true, some language models include these special symbols. We guess that the Dusty Keys developed by Chicagologic [11] is a good example for including these symbols in modelling, though neither any explicit technical information is available in the site nor any extensive research has been done to verify this.

The nature of application determines the characteristics of the text entered by the user. For example, the text entered in a word processor will be different from the one entered in an email client or the one entered in messaging editor. Grinter [1] discusses the vastly changed nature of texts entered by the users in mobile phones. They report that the users shortens the words in messaging, which many socio-linguists fear that will lead to poor spelling abilities and language as a whole among the users. Mostly this word shortening is done by a) removing the vowels (Eg. *cnt* for *can't*, *rd* for *read* etc.), b) using similar sounding character (Eg. *4* for *for*, *u* for *you*, *2morrow* for *tomorrow* etc.) for a word/ syllable and c) acronyms (Eg. *BYKT* for *But, you knew that*). While studies has been reported about shortening of words for English [1], Finnish [4], Norwegian [5] and others, no study has been undertaken for Indian languages. This is probably because of the fact that messaging in Indian languages is being introduced only recently.

We believe that, word shortening may not happen in Tamil as it happened in English, because of the fact that a vowel joins with a consonant to become a consonant-vowel in all word positions, except in the beginning. If one omits the vowel, the word will have more consonants and ultimately the text might be unintelligible. We further believe that the usage of similar sounding characters for actual one may also not occur in Tamil, because Tamil is phonetic by nature, unlike English and other European languages. However, all these observations are to be verified statistically after the introduction of Tamil messaging and we might be wronged by the creativity of the people.

However, we foresee Tamil text messaging polluting the language in the following ways. i) Mixing of more English and other foreign language words than in common usage today, ii) mixing up of the spoken form in the written language, which might blur the boundary between the spoken and written Tamil forms. All these discussions affirm that a standard

corpus may not fully capture the language model, especially when intended for a new medium like messaging.

3. Texting in Mobile - State-of-the-Art

We present here three methods of text entry on a mobile keypad, viz. i) multi-tap, ii) two-key and iii) predictive text input. In multi-tap approach, a key is pressed repeatedly few times till the desired character is entered, which follows an order. So, if one want to type *car*, the person has to type 2 three times, wait till the current active key times out and type 2 once followed by typing 7 thrice. Here there are two problems, the user has to repeatedly type a particular key, and if the next character happens to be from the same key as the current one, then the user is forced to wait till the timeout period (normally about 1.5 seconds) ends. However, some mobile phones provide mechanisms to kill the mandatory wait. In this method the no. of *Key Strokes Per Character (KSPC)* is higher, which reduces the speed- normally expressed as words per minute in the literature. Two-key approach involves two key presses for each character. The first key press selects the group, in which the character is present, while the second press signifies the position of the character within the group finally selecting the character. For example, to type *c*, the user first presses 1 thus selecting the group *abc* and then presses 3 to select the third character *c* in the group. It can be observed that two-key method requires *2l* key strokes per word, where *l* is the length of the word.

In Predictive text input, the system predicts the subsequent characters in a word, based on a series of currently typed characters. Each character requires just one key press and the prediction algorithm scans through the list of possible characters that is appropriate for the current position based on past evidence. It uses a dictionary obtained by thorough statistical analysis and language modelling and usually contains different types of data such as high frequency words, digrams and trigrams. If the word is ambiguous, series of words having the same key-sequence can be accessed using the * key in most of the mobile phones, which functions as NEXT key. Predictive algorithms generally work at the word level; meaning that they do not guess subsequent words based on available data and only try to predict the remaining characters in a word.

T9 developed by *Tegic Communications* [12] is probably the first example of predictive text input that has been put in use commercially. Silfverberg [7] has conducted detailed experiments to study the text-entry speed using the above three approaches using the model based on Soukoreff [8]. The study concludes that T9 is significantly higher then the other two approaches. It reports the speed of text entry to be 22.5, 25 and 45.7 *words per minute (wpm)* respectively for the three methods. *eZiText* developed by *Zicorp* [13] improves upon T9 and attempts to predict the next word based on the POS information. No detailed performance evaluation has been done for the eZiText based text entry, as has been done for T9 and other approaches.

Eatoni Ergonomics [14] has developed *WordWise* that simplifies the disambiguation by additionally introducing a shift key (as a meta key). This shift key is linked to one character in each group, thereby reducing the ambiguity space and the ambiguity is resolved for the rest of the keys by using predictive algorithms such as T9.

4. Implementation

4.1 Character Placement in the keypad

We have extensively analysed the standard 12-key telephone keypad that is being used for English and Hindi and have come up with the keypad design for Tamil. Tamil has 12 vowels, 18 consonants and 5 Grandtha characters bringing the total to 30. We have left out characters 'ஃ' *q* and 'ஸ்ரீ' *sri* from this list basically because, these characters do not form complex characters (Consonant vowels) in combination with other characters. However, this can be added as special characters probably along with punctuation marks.

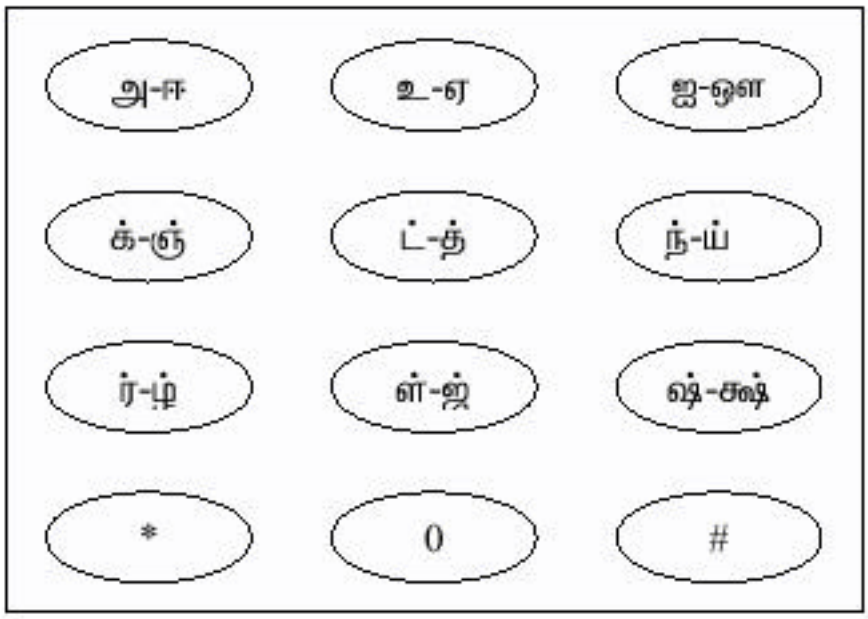

Figure 1- Proposed Tamil Keypad Layout for Mobile

Of the 12 keys, we have placed these 30 characters in 9 keys, 1 through 9, with each key being shared by 4 characters except for one key 5, which has 3 characters, see **figure 1**. We decided to retain the alphabetical order in the keypad, as in the English and Hindi keypads. The *, 0 and # keys, which are used for special purposes are left untouched in keeping with the standard. Of these, the key 0[1] doubles as 0 as well as space key, depending on the mode of operation. #[1] key chooses the cycles the mode of typing between the three options- capitalised, normal and dictionary. The *[1] key in predictive text input mode is used to navigate the list of possible words having the same keystroke sequences and in other modes it is used to select the punctuation symbols.

For typing a consonant vowel such as *ka*, the user is expected to type the corresponding consonant, *k* in this case; followed by the vowel *a*. This we believe will make the typing simple for, i) it follows the natural Tamil rule of generating a consonant-vowel by adding a vowel with a consonant and ii) it avoids the requirement of separate glyphs for each of the vowels as used in most of the font-based encoding schemes. Since, the glyphs are completely avoided, we also cut down the total no. of characters and glyphs by about 10. This implies

[1] *, 0 and # keys has slightly differing roles in different mobile phones and the information given here is as applicable to mobile phones manufactured by Nokia (http://www.nokia.com).

that even in the multi-tap approach, it takes only a maximum of upto 4 presses of the same key to get a particular character.

It should be contrasted here that, the Hindi keypad design (as available in several models of Nokia handsets) has varying no. of characters in each key ranging from 7 to 9. In the worst case, it means that the user is forced to press the same key for 9 times for typing a specific character. This complexity arises because of larger character-set of the Hindi. Another Hindi keypad design by Zicorp [13], as illustrated in its demo page [15] retains the glyphs in a separate key, which we believe is unnecessary because it fails to exploit the phonetic nature of Indian languages, apart from complicating the system.

4.2 Text Prediction Methodology

Our text prediction algorithm uses a combination of resources for predicting the text input such as most frequent words, bigrams etc. We have used the CIIL (Central Institute of Indian Languages) corpus of about 3 million words for our analysis and for the development of language model. Our analysis proved the well-known 80-20 phenomenon that the top-20% of the words in a language covering upto 80% of the language. We found that top-10% of the most frequent words cover about 81% of the corpus and top-20% covering about 87%. So, we consider the top-10% of the most frequent words from corpus for the predictive dictionary. This list can be reduced according to the memory constraints of handhelds. The statistics are given in **Table 1**.

Total no. of words	3 million
Total no. of unique words	424036
% of corpus covered by top-10% words	81.83
% of corpus covered by top-20% words	87.63

Table 1 – Corpus Analysis Statistics

We further collected the N-gram information for the 40,000 most frequent words and to our surprise found only around 680 unique bigrams. This signifies that if we use only the bigrams as the basis for our prediction, the system will be very compact and hence can be easily incorporated in a mobile device. Our trigram analysis found 5800 unique trigrams.

We also calculated the total no. of ambiguous keystrokes for our keyboard schema, shown earlier in Figure 1. The idea is to trace the keystroke sequence of each word and to find whether, any keystroke sequence is ambiguous, i.e. same sequence corresponding to two or more words. We found that about 40% of the words among those in the predictive dictionary to be ambiguous, that share about 5600 different keystroke sequences. Our further analysis are summarized in **Table 2**, and it can be seen that on an average, an ambiguous keystroke sequence has close to 3 words that has the same sequence.

The results of this analysis does not compare partly with the results suggested by Silfverberg [7], who reports that only about 5% of the words are ambiguous for English, when tested for 9025 words. We reason that, this difference is due to the highly inflectional nature of Tamil. This results in several forms of the same word that differ only by one or two characters, which might actually share the same key. For example the words பதிப்பகம் (*paTittAL*) and பதிப்பகன் (*paTittAn*), differ only in the last character and it can be seen from our keypad design that, these characters *L* and *n* share the same key 8.

No. of ambiguous occurrences	16650
% of ambiguous occurrences	40
No. of unique keystroke sequences	5653
Average no. of words per keystroke	2.94
Max. no. of words sharing the same keystroke sequence	36
No. of words requiring 10 or more presses	916

Table 2 – Results of the analysis on ambiguous keystrokes

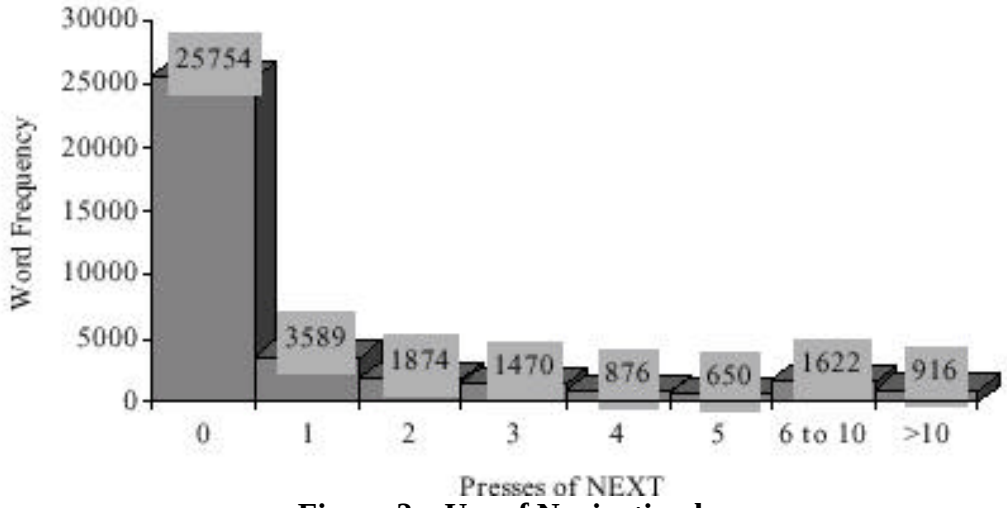

Figure 2 – Use of Navigation key

Fig. 2 above illustrates the importance of the navigation key(s). Out of more than 40000 words, about 3589 words (8.46%) requires about single press of the navigation key. The number of words requiring 2, 3, 4 and 5 presses of navigation key is 1874, 1470, 876 and 650 respectively. 1622 words require between 6 and 10 presses and 916 words require more than 10 presses.

Currently, in most of the mobile handsets, the * key is used to navigate through the list of ambiguous words. This restricts the navigation in only one direction. Here, we propose that, navigation can be made user-friendly by introducing *bi-directional navigation*, which will be functionally similar to the *back* and *forward* buttons in a web browser. So, if the user knows an ambiguous word to be of low frequency, (s)he can navigate the list from the other direction, which will reach the word mush faster than it will in the original direction. We further propose that, this can be accomplished by using either the *back-forward* arrow key-pairs or in case of their absence *up-down* arrow key-pairs; as these keys are *not* used in the predictive text input mode.

4.3 Auto completion

We have also implemented auto-completion that completes a word based on the partial information available so far, provided the keystroke sequence is unambiguous. However, the frequency information can also be used to complete the word even if it is ambiguous, such that the most frequent word is used in auto-completion. In case, the user tends to type a different word, (s)he can continue to type the word even after auto-completion and in this case, the auto-completed word will not be used.

4.4 User Dictionary and Continuous learning

The system supports the creation of user-defined dictionary, which enables a user to add custom words to the dictionary. Also, the system stores all the words that are typed by the user, which allows the system to continuously learn new words and update its dictionary.

4.5 Automatic Disambiguation

In this automatic disambiguation, we propose to automatically disambiguate the current word based on the previous word and its Parts-Of-Speech (POS) information. For example, consider an adverb வேகமாக (*veekamaaka*), followed by keystroke sequence 6-3-4-1-5-1, corresponding to words மோசடி (*moocaTi*) and போகாது (*pookaata*). Then the system automatically chooses the word as *pookaata*, which is a verb, since an adverb is always followed by a verb and not a noun, though in this case *moocaTi* occurs frequently. It should be noted that current version does not support this feature.

5. Conclusion

We present an intelligent predictive text system for Tamil texting in the mobile platform that has several sophisticated features including auto-completion and user dictionary & continuous learning. Our solution includes a simple yet powerful keypad design for Tamil input in mobile devices that will be suitable for multi-tap as well as predictive text input techniques. We also propose an innovative *bi-directional navigation* methodology, instead of the current uni-directional approach, for navigating the ambiguous words list in the context of predictive text input.

References

1. Grinter, R.E. and Eldridge, M., *y do tngrs luv 2 txt msg?* In Proceedings of Seventh European Conference on Computer-supported Cooperative Work ECSCW '01, (Bonn, Germany, 2001), Dordrecht, Netherlands: Kluwer Academic Publishers, 219-238.
2. Grover, D. L., King, M. T., & Kuschler, C. A. (1998). *Reduced keyboard disambiguating computer*. US Patent 5818437. USA: Tegic Communications, Inc., Seattle, WA.
3. ISO/IEC 9995-8. *Information systems – Keyboard layouts for text and office systems – Part 8: Allocation of letters to the keys of a numeric keypad*, International Organisation for Standardisation, 1994.
4. Kasesniemi, E. L. and Rautiainen, P. *Mobile Culture of Children and Teenagers in Finland*. In Katz, J. E. and Aakhus, M. eds. Perpetual Contact: Mobile Communication, Private Talk, Public Performance, Cambridge University Press, Cambridge, England, 002, 170-192.
5. Ling, R. and Yttri, B. *Hyper-coordination via mobile phones in Norway*. In Katz, J. E. and Aakhus, M. eds. Perpetual Contact: Mobile Communication, Private Talk, Public Performance, Cambridge University Press, Cambridge, England, 2002, 139-169.
6. MacKenzie, I. S., & Soukoreff, R. W. *Text entry for mobile computing: Models and methods, theory and practice*. Human-Computer Interaction, 17, 147-198. 2002.
7. Silfverberg, M., MacKenzie, I. S., & Korhonen, P. *Predicting text entry speed on mobile phones*. Proceedings of the ACM Conference on Human Factors in Computing Systems - CHI 2000, 9-16. New York: ACM. 2000.

8. Soukoreff, R. W., & MacKenzie, I. S. *Theoretical upper and lower bounds on typing speeds using a stylus and soft keyboard*. Behaviour & Information Technology, 14, 370-379. 1995.

Websites

9. http://ei.cs.vt.edu/~cs5724/g1/
10. http://1976design.com/blog/archive/2004/09/07/link-presentation-fitts-law/
11. http://www.chicagologic.com/default.asp
12. http://www.t9.com/
13. http://www.zicorp.com/
14. http://www.eatoni.com/
15. http://www.zicorp.com/textinputdemos.htm