Word Prediction for Tamil SMS

A G Ramakrishnan
Department of Electrical Engineering,
Indian Institute of Science, Bangalore, India

Abstract :

The goal of the project is to design all the aspects necessary for a viable, easy and efficient input method for SMS in Tamil language on mobile phone keyboards. This involves a judicial mapping of the Tamil alphabets to the number keys, creating a table of possible words and their frequencies, a GUI to test the system on the PC and to develop or use other necessary tools. Since the field is relatively unexplored, particularly for the language domain in consideration, we have also developed most of the other tools as the existing tools are not extensible. A complete solution has been simulated to facilitate users to create an SMS in Tamil on mobile phones. A character mapping to the mobile keypad has been designed, keeping in view user convenience, optimizing on the effort required. It combines the features of Tamilnet99 keyboard input standard and word prediction based on statistics of occurrence of Tamil words based on analysis of a 32 million word Tamil corpus.

Introduction

Today, in India, the number of mobile phones and the communication traffic in them have overtaken the corresponding figures of landline telephones. SMS through mobile phones have also become ubiquitous, and together with e-mails, have already created the danger of an "end" to personal communications by paper through the regular postal system, at least by the educated, city-bred and information technology-savvy population. To match these developments, it is imperative to have the facility to enter, send, receive and read e-mails and SMSes in Tamil. E-mails in Tamil are already a reality. This paper is therefore, we believe, the first attempt to come out with a solution to not only send SMSes in Tamil, but send it as comfortably as one sends an SMS in English.

Keyboard Mapping

Mobile Phone Keyboards have a limited number of keys (roughly 10). However, the use of
the mobile phone as a messaging instrument (using SMS) has risen in the recent past. Currently, predictive texting schemes exist for English and many European and other languages. However, such a scheme does not exist for most Indian languages, including Tamil. Moreover, the mobile phone keyboard for the English Language is generally arranged in the alphabetical order, that is the key for the number 2 contains the letters A,B,C; that for number 3 contains D, E, F; and so on. Following a similar arrangement is a relatively inefficient scheme for Tamil; hence, we have come out with a more efficient arrangement that does not club frequently used letters on the same key.

Our mapping assigns, in general, one vowel and three consonants to each key. Consonants that co-occur have been assigned to the same keys, which we think will reduce the effort in keying in words. A word lexicon that contains the words followed by their frequency tag (its frequency in the corpus used), is used at the back-end for the prediction. A morphological analyser has also been designed for Tamil. Based on considerable experimentation, and investigation of the results, we shall decide as to whether a morphological analyzer needs to be incorporated to provide us with better results. That is, instead of tagging just the word, we could tag its root word as well.

GUI Design

For the purposes of experimentation, we have developed a GUI that simulates the mobile phone keyboard. The GUI is programmed entirely in Visual Basic. The software provides an area for input, where the users can enter numbers and use arrow keys to move forward and backward. Alternately, the users can use the mouse for input. The GUI supports two modes, as available in most mobile phones: the smart mode, and the "a-i-u" mode. The smart mode is the default mode. In this mode, the user needs to press the key only once, and the predictive texting algorithm will guess the most probable word and the next choices. The "a-i-u" mode is the alternative mode, where the user may have to press a key a specific number of times to obtain a given alphabet. In both the modes, the user does not have to press the right arrow key to move to the next character. It happens automatically, if he/she waits for a few (currently 5) seconds (programmable). To handle both the modes, we have designed a Finite Automaton that converts the raw input (the sequence of user key-strokes) to a more 'processable' form.

Prediction Algorithm

We have used the Emille Monolingual Tamil Corpus for the frequency tagging. The Emille corpus is encoded in Unicode font, and to read this, we first convert it to Romanized Tamil. Romanized Tamil is convenient and acts as a good middle medium between the Tamil Script and Unicode. For Romanized Tamil, we are currently using the Wx notation. The encoding scheme is given in a separate input file. Thus, the code can act as a generic font converter too, if provided with the encoding scheme. The corpus is tagged and pairs of words (in Romanized Tamil) and their frequencies are stored in a separate file. Internally all the processing of Tamil words happens in Romanized Tamil, and only at the stage of the final output does the conversion to Unicode characters happen. The dictionary has been created to reside at the back-end, which simply contains each word, followed by its frequency of occurrence in the corpus. Our algorithm for predictive texting first generates all the possible combinations from the input, then compares them with all substrings in the dictionary of the same length, and screens them based on a user-defined frequency cut-off.

There is an issue to be considered here. Any Tamil input is converted so as to conform to the Tamilnet99 keyboard driver standards. Three important rules are: any consonant followed by a vowel is combined with the consonant itself, that is, ka + u will become ku;

if any consonant that is repeated twice consecutively, a diacritic is added automatically to the first one, that is, ka+ka becomes kka; and consonants preceded by homorganic nasals become soft, that is gna + ka become gnka. These rules prevent us from doing a direct search on a dictionary as a case might arise when the user enters the digit corresponding to 'ka' twice. We cannot simply continue with a set of words that end with just 'ka', because we might lose words which end with 'k'. Thus, we have to use a look-ahead to solve this problem. However, in this case, it can be observed that it is sufficient to retain just the root of the consonant; that is, instead of ka, we retain "k", and then screen out words which do not end with "k". The set of Tamilnet99 rules is easily configurable, as they are stored in a separate file.

A file contains the number-to-alphabet mapping, which can be changed for experimentation purposes. We are using an incremental algorithm to generate all the possible combinations. Since frequency cut-offs will reduce the number of words at any stage, this algorithm works considerably faster than a brute force algorithm. The word combination with the highest frequency is selected to be displayed automatically. The user can change the word by selecting a different word or spelling out the word. The words are displayed in Tamil Script using a Unicode Font. The words are displayed in Romanized Tamil as well.

Results:

We have now obtained the revised Emille corpus, and we assume that most of the errors in the initial corpus have been eliminated. By using the revised corpus, and by analysing the entire corpus, the performance of the system has improved. Though the system predicts common words, the performance of the system depends on the corpus used. We have picked disparate areas of the corpus, such as, politics, sports, cinema, etc. for the frequency tagging.

With this project, the other tools we developed could be used in other contexts as well, with little or no change. A generic font converter could be developed. The code already handles Romanized English to Tamil Script. This would be helpful for people who do not know Tamil, want to type in Tamil. Moreover, the Predictive Texting Scheme could be applied in Word Processors, as usage of Tamil in Word Processors is becoming increasingly abundant.

The project could easily be extended to other Indian languages, as it does not assume anything about the language. The requirements are a set of Unicode rules, a corpus (or a word-list with frequencies) and a set of keyboard processing rules (optional). The number-to-key mapping is easily configurable, and can be experimented with to come up with an efficient mapping.

Directions for future research are in the areas of predictive texting with syntactic analysis, and next-word-prediction with semantic analysis.