Frequency Based Predictive Input System for Hindi

N Joshi
Apaji Institute
Banasthali University
Rajasthan, India
+91-9414277014
nisheeth.joshi@rediffmail.com

I Mathur
Apaji Institute
Banasthali University
Rajasthan, India
+91-9785377807
mathur_iti@rediffmail.com

S Mathur
Syncamore Software Solutions
Maharashtra
Pune, India
+91-9819309575
shrutimathur19@gmail.com

ABSTRACT
In this paper we discuss a novel approach to input text in Hindi. Since almost all Computer users are acquainted with English and are unfamiliar of using Hindi onto the same, it creates a problem when one is required to draft a document in Hindi. We have developed a tool which addresses this problem. A user can input text in Romanized Hindi which is converted in Hindi.

Categories and Subject Descriptors
I.2.7 Natural Language Processing – language generation, language parsing and understanding, text analysis.

General Terms
Human Factors, Standardization, Languages, Verification

Keywords
Hindi, Text Input, Transliteration, Word Prediction, Edit Distance, Local Language Computing.

1. INTRODUCTION
India enjoys a very rich linguistic and cultural diversity. It is a country, where after every few kilometers the language changes. We have more than 200 living languages/sub-languages being spoken in India. Hindi is the most widely spoken language in India, fourth most popularly spoken language in the world [1]. Moreover, in India it enjoys the status of Raj-Bhasha. It is also the official state language of most of the states of India, which come under the Hindi speaking belt. Moreover, general government communication, inter-office memos are written and circulated in this language. Hindi has 46 Characters (13 Vowels (वर) and 33 Consonants (यंजन)). Vowels change their shape when are combined with consonants and can be termed as vowel symbols, moreover, some of the consonants can also be transformed into symbols (For example 'र' can have two distinct symbols Eg. as in 'tra':' ' complete 'त' and complete 'र' and 'rta':'त' complete 'त' and half 'र्').

A lot of effort has been done to linguistically unify India. The work is on to develop tools which can translate the text in one language to another language (at least in 22 Constitutional Languages of India) and to retrieve information in one's local language. But, little work has been done to provide compatibility to the urban population i.e. If a person who is not familiar in using Hindi on a computer (although he can read and write in Hindi) and wish to use the same, then he would either have to type in Romanized Hindi or would be using some Transliteration Tool and would have to remember the key in sequences which is again a very daunting task. We have developed a tool which can help a person type in Hindi with ease, without remembering what sequence to use.

This tool would not only help in providing an effect input mechanism, but also would be useful for converting text which is entered in Romanized Hindi. This tool can also be used to develop web pages content in Hindi.

2. RELATED WORK
There has been some effort on the Indian Language Computing Front to develop tools for Hindi/Indian Languages. Government of India has put in Crores of Rupees to develop tools for Indian Languages. Training programmes have been conducted to make users familiar with Local Language Computing (LLC) [2]. Most of the business sector has also put in funds to develop tools in LLC. Microsoft [3] and IBM [4] have developed tools like Input Method Engines to input text in one's own language. Moreover, various fonts have been developed to cater to the needs of an end user. In this section we discuss some of the commonly used input mechanisms.

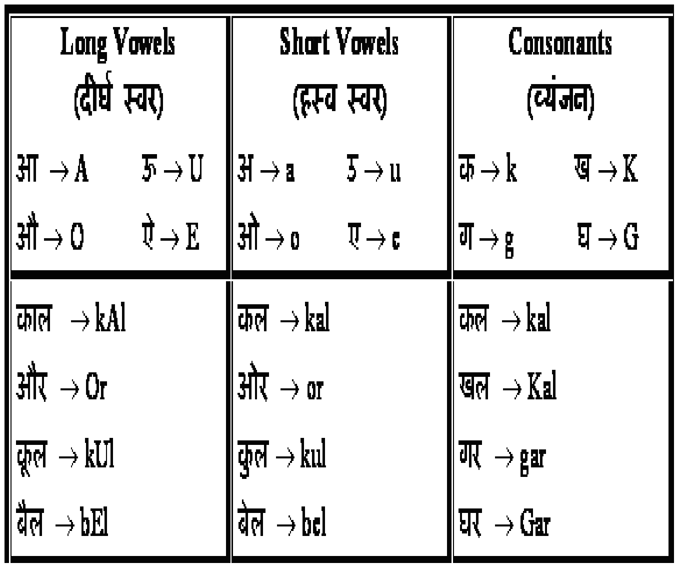
2.1. Direct Input Method
Various Fonts and Codes have been developed to support Local Language Computing. Keyboard Layouts like KrutiDev were developed [5]. Here each key on a keyboard is mapped with a Hindi character/symbol. The problem with this kind of mapping is that one has to remember which key is mapped to which character. It becomes almost impossible for a naïve user to input text in Hindi and therefore has to rely on others to input text for him.

As all the keys are assigned an ASCII/ISCII value, we cannot directly provide web pages in this format, as there is a possibility that one does not have the desired font to view the content of the website. So, when one clicks on the Hindi version of the site, he gets some garbage output, until he downloads the proper fonts provided to view the content. An alternate to this is that, one can also develop web pages using Unicode, but this still requires remembering which key is mapped to which character/symbol. This solves our problem as any page developed in Unicode does not require any additional font to be install on the user's machine, but still developer of the content have to remember key mapping. To resolve this problem, Transliteration Based Input System was developed.

2.2. Transliteration Based Input System

Transliteration is the process for writing a text of some language/script through some other language/script. For Example we write 'namaste' in Romanized Hindi which is transliterated to 'नम ते in Hindi.

There are number of tools available from internet which can be used for the purpose. Some of the prominent ones are Google Indic Transliteration, Qualipad, GIST Transliteration Utility. Out of these some are web based and some can be downloaded on your machine for offline transliteration. Broadly we can classify Transliteration Systems into two categories. They are Simple Letter Substitution Systems and Phonetic Based Systems.



Long Vowels (दीर्घ स्वर)	Short Vowels (ह्रस्व स्वर)	Consonants (व्यंजन)
आ → A ऊ → U	अ → a उ → u	क → k ख → K
औ → O ऐ → E	ओ → o ए → e	ग → g घ → G
काल → kAl	कल → kal	कल → kal
और → Or	ओर → or	खल → Kal
कूल → kUl	कुल → kul	गर → gar
बैल → bEl	बेल → bel	घर → Gar

Figure 1. Simple Letter Substitution System

2.2.1. *Simple Letter Substitution Systems*

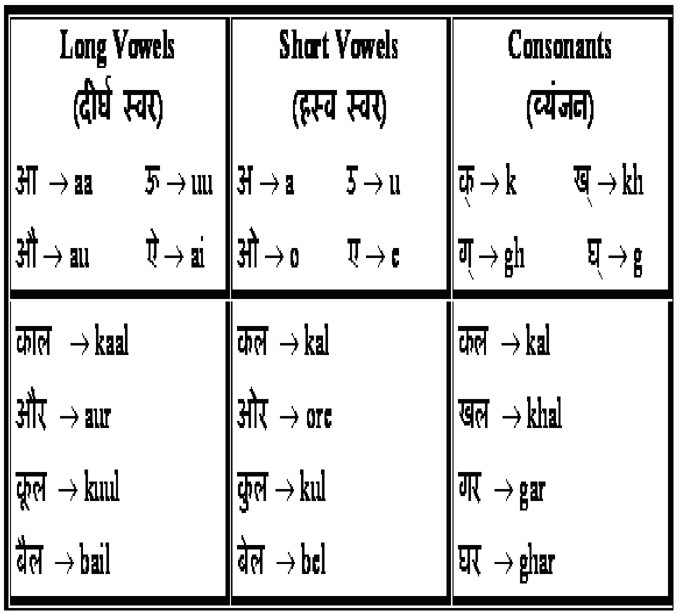
Simple Letter Substitution Systems are systems which can substitute a letter in the target text with the source text. As compared to the direct input method, here the user can more naturally write the text in Romanized Hindi so that, without learning which key is mapped to which character, one can type text more conveniently. But then again this gives rise to another problem. Unlike in English, we can characterize vowels in Hindi as Short Vowels (ह व व्रand Long Vowels (द घ)वर which create a problem as using this mapping we cannot differentiate between Short and Long Vowels of same character. Figure 1 explains the problem in more detail. Here in order to differentiate between 'उ' and 'ऊ' or 'ओ' and 'औ' we have considered upper case and lower case letters as implemented by WX Notation [6]. Here 'उ' is mapped with 'u', 'ऊ' is mapped with 'U', 'ओ' is mapped with 'o', 'औ' is mapped with 'O' and so on. For Example: Hindi Text – 'मुझे घर जाना है' can be written as 'mujhe ghar jana hai'

But, the problem with most of the Hindi speakers is that, they wish to write words as they speak, again remembering which is a long vowel and which is a sort vowel is too troublesome. More over even if they do have a knowledge and can remember mapping, regularly pressing a shift key to write text is not a very user friendly approach.

This system eludes the beauty of the language as one has to write certain words which might not resemble while being spelled out. Thus, these systems are very less efficient.

2.2.2. *Phonetic Systems*

Phonetic Systems are systems which work on the principles of Phonetics, in a laymen's language, these systems are used to simply write words as they are spelled out based on the voices/sound generated while uttering them. These systems are far more popular then Simple Letter Substitution Systems. Figure 2 illustrates this fact.



Long Vowels (दीर्घ स्वर)	Short Vowels (ह्रस्व स्वर)	Consonants (व्यंजन)
आ → aa ऊ → uu	अ → a उ → u	क् → k ख् → kh
औ → au ऐ → ai	ओ → o ए → e	ग् → gh घ् → g
काल → kaal	कल → kal	कल → kal
और → aur	ओर → ore	खल → khal
कूल → kuul	कुल → kul	गर → gar
बैल → bail	बेल → bel	घर → ghar

Figure 2. Phonetic Input System

This technique is far more intuitive then the ones discussed so far and are more suitable in South East Asian perspective where we have the concept of long and short vowels or where an English character can be mapped to multiple different characters in the local language. Here in order to distinguish between two phonetically similar characters, we can double the characters in the case of long vowels and add one or more letters to the base character for identification of a target consonant. Here 't' is used for 'त्' and 'ta' is used for ' '. For Example if we wish to write

'मुझे घर जाना है' in Hindi then we have to write 'mujhe ghar jaanaa hai' in Romanized Hindi.

This technique has been used in mostly all the transliteration tools available for Indian languages. Itrans [7] is one of the popular encoding schemes based on Phonetic Transliteration and the tool is freely available over the Internet.

In spite of all the advantages this technique too suffers from some basic flaws as different people tend to spell differently. For example, 'निशीथ' can be spelled out as nisheeth and nishith, i.e. short vowel is used where long vowel should have been used. No system discussed so far is capable to handling these kinds of problems.

3. FREQUENCY BASED PREDICTIVE INPUT SYSTEM

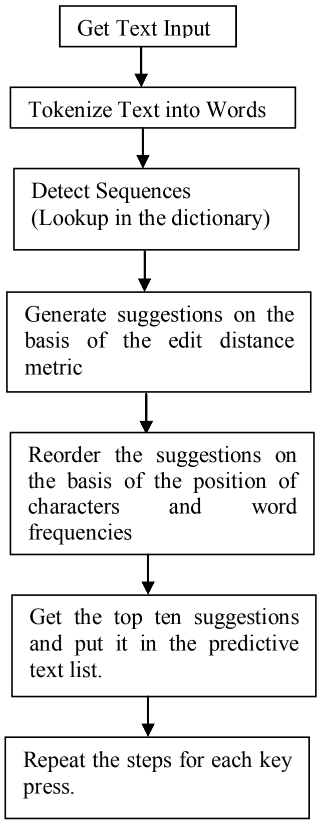

Get Text Input
Tokenize Text into Words
Detect Sequences (Lookup in the dictionary)
Generate suggestions on the basis of the edit distance metric
Reorder the suggestions on the basis of the position of characters and word frequencies
Get the top ten suggestions and put it in the predictive text list.
Repeat the steps for each key press.

Figure 3. Working of Predictive Input System

We have developed an input system which is based on phonetic transliteration and which is susceptible to word errors and tries to predict words based on the sequence of Roman characters provided by the users. In order to develop this system we created parallel aligned corpus (dictionary) of 10,000 words from tourism domain and assigned frequencies on the basis of number of occurrences of a word in the corpus. Using this system a user can type a word as per his knowledge and the system would try to guess the desired sequence of words. For Example: a user wishes to input 'भारत' for this he may choose to enter 'bharat' 'bharath' 'bhaarat' 'bhaarath' 'bharata' 'bharatha' 'bhaarata' 'bhaaratha'. This would be done on the basis of the words available in the corpus. Moreover, while designing the corpus we ensured that all the possible word sequences for a word can be registered. Once a user enters a character that character is searched on a Character Table and top ten matches to this character is displayed in the word list. This process is repeated for each keystroke. The search in the corpus was done using Edit Distance algorithm [8]. Generally when we need to write a text of some other language then basic factors like insertion, deletion, transportation and substitution are considered for text conversion. We have done the same in our system.

We have used partial match search algorithms to reduce search efforts. Edit distance metric forms the basis for ranking of suggestions in the word list. Figure 3 shows control flow of our system.

4. EVALUATION

We evaluated our system for user friendliness and efficiency. Our experiments were based on similar experiments by Goonetilleke et al. [9] which were performed on Shinalese. User Friendliness was calculated on ease of remembrance of words.

We asked a group of 60 people to write English equivalent of Hindi characters. The users were given a two column list, where in one list there was a Hindi character and in the other the user had to fill in English alphabet(s), based on their knowledge. Then we provided about an hour training to the users and gave sample exercises for Wx Notation and Itrans Notation. Next, we gave two column list for both these schemes. We calculated edit distance based on each given encoding scheme, we calculated the given edit distance the user's transliteration and the scheme's transliteration.

The results are presented in Table 1. As shown, our predictive input system performed better the the two input schemes.

Mapping Scheme	**Average Edit Distance Per Word**
Wx Notation	0.90
Itrans	0.59
Predictive Input (Our Work)	0.34

Table 1. Edit Distance Experiment

Here average edit distance was calculated based on the following formulae:

1. Avg_edit_dist(char) =

$$\frac{1}{\#Subs}\sum_{subs=1}^{\#Subs} edit\_dist(inp\_seq(char), proposed(subs, char))$$

2. Average =

$$\sum_{Char=1}^{\#Char} freq(char) \times avg\_edit\_dist(char)$$

We calculated average edit distance between input key sequence and proposed text of each character. The average edit distance was calculated using the above equations. The results show that there is big difference between our system and WxNotation. Our average edit distance was far lesser then this input system. On the other hand difference between Itrans and our system is very small. This is because people onto whom these tests were conducted tried to generate Romanized Hindi word which resembled English word i.e. they tried to avoid long vowel combinations like 'ii', 'uu', 'aa'.

5. CONCLUSION AND FUTURE WORK

In this paper we proposed a scheme for Text Input for Hindi. This can further be extended to non roman script based languages. In this approach we used large corpus to develop our system. The efficiency of our system is directly based onto the word frequencies. This system provides the list of predictive text almost instantaneously.

As an extension to this work, we plan to test the system with different domains of corpus. We also plan to test the system with some other more efficient algorithms and devise a better ranking system which is free of frequency based estimation. Moreover, we also plan to put the system on the Internet so that it can provide input to the users on the fly.

6. REFERENCES

[1] Bentley J.L. and Sedgewick R., Fast algorithms for sorting and searching strings, In Proc. ACM-SIAM Symp. On Disc. Algorithms, 1997.

[2] Bharati A., Chaitanya V. and Sangal R., Natural Language Processing: A Paul Lewis M., Ed., Ethnolouge: Languages of the World, Sixteeneh Edition. SIL International.

[3] IT Localization Clinics. Viswabharat, Jan 2001.

[4] Introduction to Indic Languges: Globalize you ebusiness. http://www-01.ibm.com/software/globalization/topics/indic/storage.jsp

[5] Joshi A., Ganu A. et al, A keyboard for text entry in Indic scripts. In: Proc. Computer Human Interaction, 2004.

[6] Paninian Perspective, Prentice Hall of India, 1999.

[7] Report of the Committee for Standardization of Keyboard Layout for Indian Script Based Computers. Electronics Information & Planning Journal, Vol 14, No. 1. October 1986

[8] Sandeva G., Yoshihiko H. et al, SriShell Primo: A Predictive Sinhala Text Input System, In: Workshop on NLP for Less Privileged Languages (NLPLPL), International Joint Conference on Natural Language Processing, 2008.

[9] Wissnik Cathy, Windows Indic Script Support. http://www.bhashaindia.com/Developers/MSTech/indicsupport/index.aspx