

LANGUAGE IN INDIA
Strength for Today and Bright Hope for Tomorrow
Volume 6 : 7 July 2006

Managing Editor: M. S. Thirumalai, Ph.D.
Editors: B. Mallikarjun, Ph.D.
Sam Mohanlal, Ph.D.
B. A. Sharada, Ph.D.
A. R. Fatihi, Ph.D.
Lakhan Gusain, Ph.D.
K. Karunakaran, Ph.D.
Jennifer Marie Bayer, Ph.D.

**TECHNOLOGY FOR INDIC SCRIPTS –
A USER PERSPECTIVE**

Renu Gupta, Ph.D.

Technology for Indic Scripts: A User Perspective

Renu Gupta, Ph.D.

Abstract

Although Indians represent a sizable market for computers and mobile phones, the technology for typing and displaying text in Indic scripts has lagged far behind the demand. The main hurdles have been (a) the nature of the Indic scripts and (b) the lack of compatibility across software providers. This paper examines three input devices—manual typewriters, computer keyboards, and mobile phone keypads—that have modified existing models for English. It compares these with the technology developed for typing text in Japanese.

1. Introduction

In 2005, when I was writing about the scripts used in India, I faced difficulties typing the paper. The paper was written in English with examples from Urdu, Hindi, and Japanese. Two word processing programs—Microsoft Word and LaTeX—could handle words in English as well as Japanese; but when I tried to add Devanagari, MS-Word did not even offer this option and LaTeX crashed. Now, a year later in 2006, the technology has advanced to the point where I can type a paper in three or more scripts; however, there is no guarantee that readers can view these scripts on their computer or on the web, which is why the paper is available only as a pdf file.

In this paper I examine the technology available for Indian scripts from a user's perspective. The paper describes three input devices for Indic scripts—the manual typewriter, the computer keyboard, and the mobile phone keypad—which are modifications of existing devices for English. These are then compared to their equivalents for typing in Japanese, where innovative solutions had to be designed to handle the complex writing system.

2. Background

Between 1994 and 1995, Internet services and mobile phones became available in India. These enabled Indians to communicate instantly across distances through the written medium, using email on computers and text messaging on mobile phones (which was cheaper than a phone call). Since the introduction of these services, the number of users has increased dramatically; from 5 million users in 2000, the number of internet users in India jumped to more than 50 million by the end of 2005 (Internet World Statistics, 2006) and the number of mobile phone users stood at 96.9 million by the end of April 2006 (TRAI, 2006).

Despite these growing numbers, resources for typing and viewing text in Indian scripts have not kept pace. Until recently, there was neither any hardware nor software to help the general user to type, display, or view texts in Indic scripts either for SMS messages or on computers (here I am excluding packages such as LaTeX that require special commands and do not have a graphical user interface). Indians circumvented this problem by typing their messages either in English or in an Indian language—but in the only script available, which was the Roman script.

Little attention was paid to Indians who needed or wanted to type in their own languages; instead, Indian software programmers focused on writing programs for English language software. In discussing the lack of ‘vernacular software’, Keniston (2001) argues that software companies assumed that the Indians who could afford computers knew English. He cites Harsh Kumar of BharatBhasha who pointed out that small-and medium-sized merchants want Indian language computers for their businesses, but in the absence of supply there is no demand. Since then, businesses have recognized the need and attention is being directed toward developing software for users to type in Indian languages on computers and mobile phones.

How many people and scripts does this involve? If we were talking about a few thousand users and a few scripts, it may not be worth the effort of developing the resources for

typing in Indic scripts. However, in India there are nine official Indic scripts—Bangla/ Assamese, Devanagari, Gujarati, Gurmukhi, Kannada, Malayalam, Oriya, Tamil, and Telugu—and several other scripts, such as Modi and Khaithi, that are used within a community. In addition, many of the scripts currently used in Southeast Asia are derived from Indic scripts and share the same organizing principle. Indic scripts currently in use include the scripts in Burma (Mon), Cambodia (Khmer), Laos (Lao), Thailand, and Tibet. Indonesia has a number of scripts derived from the Indic scripts—for example, Balinese, Batak, Lontara, Redjang, and Javanese—but with the introduction of Bahasa Indonesia as the national language, these scripts are rarely used now. However, as the technology for writing them became available, there is renewed interest in these scripts (see, for example, the website by Suweda). Other Indic-based scripts that are rarely used now are Tagalog in the Philippines and Siddham in Japan (again, see sites like *Modern Siddham* and *Bonji* for recent interest in the script). Given this vast pool of users who use scripts with a similar organizing principle, the need for the technology becomes important.

3. Obstacles to Technology Solutions

There are two main problems in designing and providing resources for users to type in the Indian scripts. The first problem is the nature of the Indian scripts and the second is the lack of compatibility across providers.

3.1 Structure of Indic scripts

The QWERTY keyboard was designed for the English script and is easily adapted to other alphabetic scripts, such as Cyrillic. Alphabetic writing systems use a limited number of characters that fit on a keyboard; using the Shift key provides 92 symbols on a standard keyboard, which is sufficient for the letters (upper and lower case) as well as numbers, punctuation marks, and special symbols. To accommodate special characters for other alphabetic scripts, such as the tilde in Spanish or the accents in French, certain keys are easily configured. In an alphabetic writing system, the letters are typed in a linear sequence.

For users of non-alphabetic writing systems, typing is not an easy matter. The alphabet is only one of six types of writing systems (Daniels, 1996); the five other writing systems are logographies (such as Chinese), syllabaries (such as Hiragana and Katakana in Japanese), abjads (such as Arabic), abugidas (such as the Indian scripts), and featural systems (such as Hangul in Korean). Users of these writing systems have had to devise ways to adapt the technology to allow them to write in their own scripts and the solutions have not always been add-ons to the available hardware and software.

While Indic scripts do not use as many characters as Japanese or Chinese, they are still listed as complex scripts. In fact, given the complexity of Indic scripts, there have been proposals to use the Roman script instead (Gogate, 2003) or to simplify the scripts (Musa, 2006a, 2006b). These Indic scripts or abugidas (Daniels, 1992) have descended from a common script, Brahmi, and hence they have a common organizing principle. In an abugida, “each character denotes a consonant accompanied by a specific vowel, and the other vowels are denoted by a consistent modification of the consonant symbols” (Daniels 1996: pg. 4). Using Devanagari as an example, the character *k* is accompanied by the vowel /ə/; to indicate a different vowel, matras are added to the character *k*, as in कि (before), कै (above), की (after), and कू (below). In certain consonants, such as र (for *r*), the *matra* is positioned in the middle of the symbol as in रू. To complicate matters, the shape of the character changes under certain conditions; for example, the shape of the symbol for *k* differs in the following: *k*, *kya*, and *kk* (namely, क, क्या, and क्क). In addition, there are ligatures such as श्री, which cannot be decomposed into their composite consonant symbols.

This property means that the multiple shapes for each consonant have to be available to the user. In fact, this is done in printing where different blocks are designed for each individual shape, but such an option is not possible on typewriters, computers, and mobile phones where the keyboard/keypad has a limited number of keys for data entry.

However, computers and mobile phones use software that can and should handle these multiple combinations, displaying the correct forms for the user.

3.2 Compatibility issues

Despite efforts by the Government of India to establish standards for computing in 1986, organizations, individuals, and vendors have developed software that is highly creative but not compatible across platforms (Keniston, 2001). This restricts the user not only in terms of typing text but also in terms of display, because readers or receivers have to use the same or compatible systems in order to view the material. With the adoption of Unicode, a standard has emerged but this is not supported by older browsers or operating systems.

4. Input Devices

The following sections describe three different input devices—typewriters, computers, and mobile phones. They are discussed in terms of what they offer for Indic scripts; each description is followed by the solution devised for typing Japanese language text. Japanese uses three different writing systems—a logography (called Kanji), two syllabaries (called Katakana and Hiragana), and an alphabet (called Romaji which is composed of English letters) –all of which are taught in school.

4.1 Typewriters

Typing in an Indian script on the manual typewriter was never easy. The standard QWERTY keyboard for English was picked up and symbols for Indian scripts were mapped onto these keys. Krishna (1991) points out that the result has been less than perfect for two reasons: (a) since the consonants and the *matras* are treated as different units, their alignment on the page is uneven, and (b) for typists, the load on the left hand is disproportionately heavy.

For the manual typewriter, the Japanese had to come up with a novel solution because hundreds of characters have to be accommodated on a typewriter. Figure 1 shows the solution devised for the Japanese manual typewriter, which is similar to type-setting. The typist operates a lever that picks up each individual character, loads it in the typewriter, and presses a lever to type the character.

Figure 1. Japanese manual typewriter



(a) Japanese manual typewriter.



(b) Levers to move the tray and pick up a character.



(c) Tray containing individual characters.

Typing on the Japanese manual typewriter is a slow process and the typist clearly has to be an expert. When computers incorporated software for typing in Japanese, these manual typewriters were quickly abandoned; they have been phased out and moved to the backrooms of government offices.

4.2 Word processors

4.2.1. Text input

To type text in an Indic script, there are two main options: to input the text using English or to input it directly through the Indic script.

(a) Typing through English

This system is used by Linux in their BolNagri keyboard as well as the software program, *Baraha*. The user types the words using the English spelling of the word. For example, to type the word *kal*, the user types k + a + l on an English keyboard and the word कल is displayed. Currently, *Baraha* offers the facility in nine scripts.

This system (sometimes called transliteration or phonetic) is extremely simple but only for users who already know English; it disadvantages those who are not familiar with the English alphabet and the location of the symbols on the English keyboard.

(b) Typing through an Indic script

The second option is a keyboard that maps symbols for an Indian language on the keyboard. The most commonly used keyboard is *Inscript* (which stands for *Indian Script*), which was designed by the Department of Electronics (DoE) and has been adopted by Microsoft. Since the organizing principle for all the Indic scripts is the same, the keyboard was designed so that it could be used for all Indic scripts. For example, the symbol for *k* is located on the same key across all Indic scripts.

Since the development of the Inscript keyboard, other keyboards for the Indic scripts have been developed (for a comparison of keyboards for Indic scripts, see the site for Sun Developer Network). In these keyboards, the characters have been organized as follows (Figure 2):

- i) Vowels and consonants are separated. In the Inscript keyboard, the vowels are on the left side of the keyboard while the consonants are on the right.
- ii) Related sounds and characters are placed on the same key making them easier to find. For example, the same key is used for the characters *k* and *kh* (क and ख); *k* is accessed in the normal mode, whereas *kh* is written with the Shift key depressed. Similarly, the vowel for *a* (आ) as well as the *matra* for *a* (ा) are on the same key.

With the exception of these two principles, the logic underlying the placement of the characters is not obvious to users; they are not organized based on the sequence in the Indic scripts, so that the beginner spends a great deal of time hunting for the keys.

Figure 2. Inscript Keyboard for Devanagari



(a) Normal mode



(b) Shift mode

There have been attempts to modify the layout of these keyboards. The Devanagari keyboard developed by researchers at IIT Bombay (Joshi, Ganu, Chand, Parmar, & Mathur, 2004) also separates the vowels and consonants, but the consonants are arranged in the Varnamala sequence, namely, क, ख, ग, घ... ($k, k^h, g, g^h \dots$). Since this is the order in which learners acquire the script, it reduces time spent hunting for keys—a point made by the designers based on their usability studies.

When typing directly in an Indian script, the user relies on the graphic shape of the symbol to input characters and does not have to go through an intermediate language or script, such as English. This keyboard also helps the multi-scriptal user who wants to type a text in two or more different Indic scripts, since the keys are identical across Indian scripts. However, since the different Indic scripts have different numbers of characters, for certain scripts such as Tamil there are a number of redundant keys on the keyboard, which slows down the typist.

4.2.2 Text Display

In addition to dealing with the keyboard layout, users have to learn to work with what Unicode offers. By typing in Baraha, I circumvented the keyboard problem but when the same text was typed in the Unicode window in Baraha or in Microsoft Word (which uses Unicode), its difficulties and limitations became apparent.

Punctuation marks

To accommodate the large number of symbols on the keyboard, the keys for punctuation marks have been assigned to symbols. The question mark, for example, is used for the symbol ऋ. (Don't Indians ask questions?). To type a question mark, one has to hold down AltGr + Shift and press the question mark key or one can switch back to English mode and type the question mark. This applies to all the special characters on the top row, such as !, @, #, etc.

Vertical conjuncts

When one types a word like *chitthi* (letter) in Devanagari, Baraha can display चिट्ठी but Unicode displays चिट्ठी because vertical conjuncts, such as ढ़, have not been encoded as Unicode characters (Technology Development for Indian Languages, 2002; Unicode Standard Version 4.0, 2003). Fonts for many of these complex conjuncts are available in sophisticated packages such as Baraha (2006) and LaTeX (Acharya, 2001) but have been omitted from the Mangal font supported by Windows. As a result, we are working with a reduced number of characters that are adequate but the quality of the output is poor.

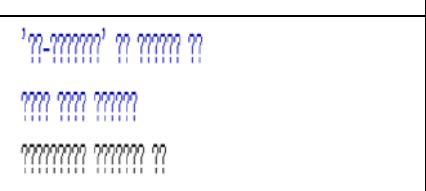
Zero Width Joiner (ZWJ)

When I tried to type my name in Unicode, I discovered the Zero Width Joiner (ZWJ). The ZWJ is a mechanism to prevent conjuncts being automatically formed. So, instead of क्ष, I can choose to write क्ख using a ZWJ for which one holds down Control + Shift + 1. When I tried to type *Gupta*, I got गुप्ता because the program treats *pt* as a conjunct. In order to get गुप्ता, I had to use the ZWJ to prevent a *pt* conjunct.

4.2.3 Viewing Text

Texts are written to be read. Determined users may manage to create messages and texts in an Indic script but what does the reader see?

Figure 3. Text display

दुबई (भाषा) : पाकिस्तान	Display on Windows using Firefox
	Display on Unix

Herein lies a major problem because readers can view the Indic characters only if their computers have the appropriate fonts. When the appropriate fonts are not available for your computer or the browser, junk characters are shown on the screen (see Figure 3).

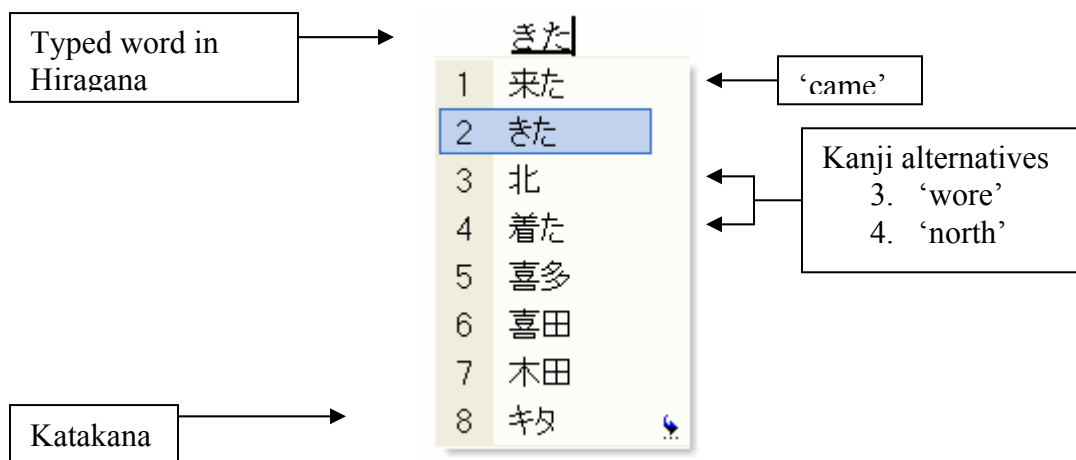
Word Processing in Japanese

With the advent of the computer, typing in Japanese has become considerably easier than using the manual typewriter. The user inputs text through a standard keyboard using either the Roman letters or one of the kana (Hiragana or Katakana) and the difficult job of kanji characters is handled by the software.

Unlike the Inscript keyboard, hunting for keys on the keyboard is not a problem even for novice users. The reason is that when software for Japanese was first created, the designers capitalized on user knowledge of Romaji, which is taught during early schooling, as well as the fact that an alphabet can be used to write a syllable, e.g., the syllable き, which is pronounced /kii/, can be written as *k+i*.

Figure 4 shows a word (*kita*) that was typed using Hiragana. To accept the word, the user presses the Return key. However, since there are many possible characters for each word in Japanese, the user can press the Space Bar to invoke an Input Method Editor (IME) that shows possible alternatives from which the user can select the correct one.

Figure 4. Typing in Japanese using an IME



For users of Indic scripts, an IME would allow them to select the appropriate character from a set. Initially, Microsoft did not consider Indic scripts complex enough to warrant the use of an IME (Kaplan and Wissink, 2003), but now a basic IME has been implemented in Windows. This eases the user's task of typing by showing a standard set of composite CV symbols -- क, का, कि, की.... (*ka, kaa, ki, kii...*) but does not offer alternatives such as complex ligatures.

4. Mobile phones

Text messaging or SMS on mobile phones is a popular mode of communication in India because of their low cost. Until the software for Indic scripts became available, Indians typed messages either in English or in Indian languages using the Roman script.

The method of text input for Indic scripts closely follows the design for English and Japanese input. In fact, the keypad interface for English, Japanese, and Indic scripts converge because the interface design is relatively recent, which allows manufacturers to learn from one another.

The mobile phone has a limited number of keys (about 10) on which the script symbols have to be accommodated. In English, each key has 3 to 4 characters and multiple presses allow the user to arrive at the required letter. However, Indic scripts and Japanese require a larger number of symbols merely for input. The keypads for Indic scripts and Japanese have a similar design—each key is used for one set of symbols and multiple taps on the key allow the user to select the correct symbol. In both cases, the sequence is based on the way the script is taught in school. In Figures 5 and 6, which show the keypads for Devanagari and Japanese, the alphanumeric key for 2 is used in Hindi to access क, ख, ग, घ, ङ (k, k^h, g, g^h, nga) and in Japanese to access the *k* sequence का, き, く, け, こ (*kaa, kii, koo, ke, ko*). (Note that among handset providers in India, there is no standard for the symbols assigned to each key.)

Figure 5. Keypads for mobile phones



Keypad for Devanagari mobile phone
(Adapted from Reliance)

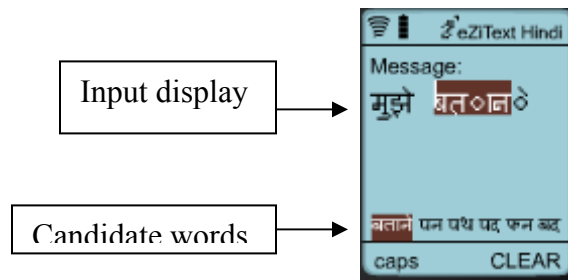


Keypad for Japanese mobile phone
(Adapted from T9® Redux)

The major difference between them lies in the amount of support provided to the user. In Japanese, the user inputs symbols in kana, which is sound-based. Since there are many homonyms in Japanese, the meaning may not be obvious from the sound (see Figure 4, where multiple meanings are possible for the string *kita*). Hence, the kana string is converted to Kanji and the candidate Kanji characters are displayed; the user selects the proper Kanji from the displayed list.

This degree of support is not necessary for typing in Indic scripts and interesting choices have been made here. Figure 6 is a screen shot of a demo for Hindi predictive software (Zi Corporation, 2006). As in the case of Japanese, the first line shows the user input while the words at the bottom are candidate words for the user to select. In the text input line, the symbols remain discrete and have not been integrated into syllables. Given the likelihood of making a mistake while tapping keys and the difficulty of moving the cursor within the small display area, if the user has made a mistake s/he can easily locate the error in such a display.

Figure 6. Text input in Devanagari on a mobile phone



(from Zi Corporation)

5. Conclusion

Since the structure of the Indic writing system is complex, it is difficult to type an Indic script on devices that were originally designed for alphabetic writing systems, such as English. This paper examined three input devices for Indic scripts from the user's perspective. Although attempts have been made to provide this facility to users, the input devices for computers and mobile phones are awkward to use, and there are still problems with displaying the text.

Several broader issues have not been covered in this paper. I have not examined one of the most important issues—interface localization—that should provide menu options in Indian languages. For issues related to Indic scripts other than Devanagari and problems in text processing, see the websites by:

- Baraha (http://www.baraha.com/web_docs/unicode_issues.htm) and
- IIT Madras (http://acharya.iitm.ac.in/multi_sys/uni_iscii.html).

Acknowledgments

The author gratefully acknowledges the following companies whose products are shown in this paper: Reliance Infocomm, T9®, and Zi Corporation. I would also like to thank the following for their help on this paper: City Hall, Aizu-Wakamatsu City demonstrated

the manual Japanese typewriter, Katsuko Kuwada arranged my visit there and translated the kanji characters for the IME, Tomoko Izumita clarified the use of Romaji in Japanese computing, and Lothar M. Schmitt demystified browser display issues.

References

- Acharya (2001). *Fonts for Indian languages*. Systems Development Laboratory, IIT, Madras. http://acharya.iitm.ac.in/ind_fonts.html. Retrieved June 5, 2006.
- Acharya (2003). Limitations of Unicode and ISCII (http://acharya.iitm.ac.in/multi_sys/uni_iscii.html). Retrieved June 5, 2006.
- Baraha. (2006). *The Glyph Codes of BRH Devanagari font*. http://www.baraha.com/web_docs/glyph_codes_dev.htm. Retrieved June 10, 2006.
- Daniels, P.T. (1992) The syllabic origin of writing and the segmental origin of the alphabet. In P. Downing, S.D. Lima, and M. Noonan (Eds.) *The linguistics of literacy* (pp. 83-110). Amsterdam: John Benjamins.
- Daniels, P.T. (1996) The study of writing systems. In P.T. Daniels and W. Bright (Eds.) *The world's writing systems* (pp. 3-17). New York: Oxford University Press.
- Gogate, M.N. (2003). A case for Roman lipi for Indian languages. *Language in India*, 3, 3.
- Internet World Statistics. <http://www.internetworldstats.com/asia.htm#in>. Retrieved May 22, 2006.
- Joshi, A., Ganu, A., Chand, A., Parmar, V. & Mathur, G. (2004). *Keylekh: A keyboard for text entry in Indian scripts*. CHI 2004, Vienna, Austria. <http://www.idc.iitb.ac.in/~anirudha/papers/ex06-joshi.pdf>. Retrieved May 22, 2006.
- Kaplan, M.S. & Wissink, C. (2003). *Unicode and Keyboards on Windows*. 23rd Internationalization and Unicode Conference, Prague, Czech Republic.
- Keniston, K. (2001). Language, power, and software. In C. Ess (Ed.), *Culture, technology, Communication: Towards an Intercultural Global village* (pp. 283-306). Albany, NY: State University of New York Press.
- Krishna, S. (1991). *India's Living Languages*. New Delhi: Allied Publishers Limited.

- Modern Siddham (2004). <http://www.mandalar.com/DisplayJ/Bonji/index.html>.
Retrieved June 5, 2006.
- Musa, S.M. (2006a). Nayanagari: A simple script for Devanagari. *Language in India*, 6, 1.
- Musa, S, M. (2006b). A simple script for Bangla and the IPA mapping thereof. *Language in India*, 6, 1.
- Reliance Infocomm. Reliance Mobile handset.
http://www.relianceinfo.com/Infocomm/Rim/handsets_lgrd2430.html.
- Sun Developer Network (2006).
<http://java.sun.com/products/jfc/tsc/articles/InputMethod/indiclayout.html>.
Retrieved May 31, 2006.
- Sudewa, I.B.A (2006). *Balinese Script Standardization and Implementation on Computers*. <http://www.babadbali.com/aksarabali/presengl.htm> Retrieved June 1, 2006.
- T9® Redux. <http://www.t9.com/japan/>. Retrieved June 5, 2006.
- Technology Development for Indian Languages (2002). Revision of Unicode Standard-3.0 for Devanagari Script. *Journal of Language Technology*. January.
<http://www.tdil.mit.gov.in/newsIndexJan02.htm>. Retrieved June 10, 2006.
- Telecom Regulatory Authority of India (2006). *Growth in telephony continues in April 2006--4.6 million subscribers added*. Press Release No. 41/2006.
- Unicode Standard Version 4.0 (2003). *South Asian Scripts*.
<http://www.unicode.org/versions/Unicode4.0.0/ch09.pdf> Retrieved June 5, 2006
- Zi Corporation (2006). *eZiText Hindi for the mobile phone demo*.
<http://www.zicorp.com/ezitexthindi.htm> Retrieved June 5, 2006
-

Renu Gupta, Ph.D.
Center for Language Research, University of Aizu
Aizu-Wakamatsu City
Fukushima, 965-85-80, Japan
email: renu@u-aizu.ac.jp