Informatics Studies. ISSN 2320 – 530X. Vol. 3, Issue 4,
Fourth Quarterly Issue. October – December, 2016. P 31-44

Computing in Indian Languages for Knowledge Management: Technology Perspectives and Linguistic Issues

Ram Awathar Ojha

Abstract

The paper is the first among a series of documents on computer applications in Indian languages to be published in the journal. Traditionally, computer applications were based on English as the medium of interaction with the system. The technology is not yet viable to Indian languages. Majority of Indian population does not speak or write in or understand English and this weakness of technology hinders the attempts to use computers for education and literacy, meant for majority of the population who should get the benefit of Information Technology. In many parts of the world computer applications have been developed in different regional languages, appropriate to the user communities. This introductory paper gives a perspective of research and development on computational linguistics, the phonetic aspect of Indian languages, transliteration standards, lack of uniformity, data entry methods, keyboards, phonetic mapping of the vowels and consonants, web pages supporting display of Indian language text etc in the context of computing in Indian languages.

Keywords: Computational Linguistics, Indian Languages, Telugu, Kannada, Phonetics, Alphabets, vowels, consonants, syllables, transliteration standards, data entry methods, keyboards, , web pages

This paper covers Writing Systems Codes for the Aksharas, The Phonetic Aspect of Indian Languages, The importance of Transliteration, Lack of Uniformity Between Different Schemes. Data Entry Methods, Keyboards, Phonetic Mapping of the Vowels and Consonants and Web Pages Supporting Display of Unicode Text**.**

Brief Introduction

The computer programs, which permit user interaction with the computer in one or more languages, where the language can be selected dynamically, either at the time of invocation of the program or subsequently during its execution is termed as Multilingual Systems (MLS) in the context of this discussion. Typically, an MLS will permit users to interact with computers in their own mother tongue. Such a system will have far reaching consequences in a country like India where English is not spoken or understood by the majority of the people living in areas away from urban environments.

Over the years, the bulk of software development in India has been carried out primarily through the English language. Knowledge of English is essential for the development of Information Systems since virtually all development packages rely on English specific input. Current software development tools can also be used to

develop an application that incorporates little or no English in its user interface. Hence MLSs are not only feasible at the level of an application program but can also present a truly localized environment for a user desiring to interact with computers in regional languages. This approach has the added advantage of providing uniformity across computer systems where, regardless of the machines like PC, MAC, Workstation etc. the user will see the same interface.

MLS catering only to the display of information are easier to design and build. The phenomenal growth of the internet has created the need for internationalization of the web where, using the right software tools, one can present information in the form in which it would be received best. However, technical challenges faced in implementing such MLS and the lack of standards have retarded the development, especially in respect of the languages of India.

In the past, MLSs in Indian languages have basically concentrated on data preparation and printing (DTP). The increasing demand for using computers in the vernacular has forced developers to look at the user interface with seriousness. The Systems Development Laboratory, Department of Computer Science and Engineering at the Indian Institute of Technology, Madras (IITM) has done unique research and development to provide a quality MLS for the country that is truly Indian in concept, design and implementation. They have developed numerous software packages to permit user interaction with computers in different Indian languages during the late eighties and they continue their work. The series of papers including the present one are intended to present before the readers the experience shared by concerned scientists of IITM in dealing with this interesting but complex problem.

Computing in Indian Languages

Computing is a general term which refers to information processing, where information is associated with some data, typically a text string, a number, an image and so on. One writes computer programs to manipulate data. Computing in Indian languages may broadly relate to computer programs which process Indian Language text strings, which may be input through a keyboard and displayed on a conventional screen display.

A primitive approach to computing in Indian languages may be through computer programs, which do string processing on texts of Indian languages much the same way it is done for the Roman (ASCII) strings. This way, many existing applications may be adapted to work with text strings in Indian languages. It turns out that this is not a simple task on account of the large set of aksharas (letters) that have to be handled. While the older techniques seem to be well suited for displaying the Indian Scripts, data entry becomes formidable. Hence, new approaches to handling Indian language text have become essential.

Often people ask questions such as 'why not have an Operating System run in Tamil or Bengali?' just as Arabic or Japanese Windows. There is no satisfactory answer to this question. In the first place, building support for Indian languages within Operating systems is not an easy job, especially when one looks for uniformity in use across all the Indian languages.

There is a general feeling among the professionals that it is best to deal with Multilingual information at the level of the user application. That is to say, keep the language aspects outside the Operating System. This way, the Operating Systems are rid of the problem of having to deal with varied character sets across many different languages. While some persons point to the success of Unicode implementation in Microsoft Windows, it must be clearly understood that the system kept away from Indian languages for an important reason. Unicode is just not right for linguistic processing with Indian Languages and is too

complex to handle even for one Indian script, much less for all the scripts in a uniform way. For all practical purposes Unicode has retained only the eight bit coding structure (actually only 128 codes) for all our scripts, which is really the bottleneck in handling the aksharas. For efficient string processing, it is necessary to work with the basic linguistic quantum of our languages, which happens to be not a letter of the alphabet but an Akshara, which is actually a syllable. The problems associated to the current implementation of Unicode for Indian languages, or for that matter the ISCII code itself, the basic standard that led to the Unicode representation for our languages will be discussed later.

Any meaningful approach to computing in Indian languages must provide for unique identification of the full set of aksharas, through fixed length codes and also provide a uniform approach to data entry of the aksharas across all the languages. Any other approach will suffer from incompatibilities between the user interfaces across different Indian languages. While solutions specific to a language may indeed be feasible, one is looking for solutions, which can be used all over the country.

For the present, and at least for some years to come, it is best to handle the problem by writing applications which handle Multilingual information directly, that is to say that the Operating System's support should not be sought as there is no consensus among the professionals on what this support should be. The total arbitrariness with which data entry in Indian languages has been handled, not to speak of the lack of uniform representation across the languages, makes it necessary for us to take a serious look at standardization. The problem is further compounded by the individual approaches taken by vendors who seem to think that the concept of the language kit with an Operating System will solve the problem.

The phonetic base and the concept of the Akshara is unique to the languages of India. It is best to deal with the problem of computing with Indian languages by first understanding how aksharas have been used in our ancient as well as modern texts. This will give us an insight into the linguistic base of our languages, allowing us to come up with a universal approach to dealing with all the languages of the country.

General Introduction to Indian Languages

Numerous languages are spoken in India. Most of them relate to one of the officially recognized languages and there are about eighteen languages identified for regular use in the country. All these languages have a phonetic base, though their writing systems vary. Some of the languages have a common script and some have scripts of their own. There are nine basic scripts besides the scripts for Urdu and Sindhi. These nine constitute the basic scripts of India. The eighteen languages mentioned above, have been given the status of official languages by the Government. Though the use of a language may appear to be confined to a region within the country, the mother tongue of many persons living in that region may be quite a different one, traceable to early migrations of families.

All the recognized languages; mostly referred to as the regional languages have a phonetic base. It is seen that there is a substantial set of words common to many of these languages and the roots of these words may be traced to specific languages such as Sanskrit or Tamil, both of which are considered very ancient languages.

Linguistic aspects of Indian languages have always attracted scholars from different parts of the world on account of the hoary past of the languages as well as their unique phonetic base. Another interesting aspect of Indian languages is the fact that language was a means not merely for communication

in daily life but also for expressing religious, philosophical, scientific and professional concepts in amazingly compact ways.

Computers and Indian Languages

It is not surprising therefore, to find renewed interest today, in studying and understanding many of India's ancient literary works. With the possibility of using computers for linguistic studies and with the increasing demand to disseminate information in the vernacular, computing in Indian languages has gained significance. Though applications such as word processing, Desk Top Publishing etc., have been successfully implemented for Indian languages, the solutions remain substantially language specific. One is happy that many of these applications are really useful in practice, in spite of the effort needed in handling data entry. Yet, very little seems to have been done in respect of electronically processing the information. Viewed nationally, there is an urgent need to provide a uniform and meaningful software solution to computing in Indian languages.

The phonetic nature of the languages leads to a writing system, which represents sounds through unique symbols. Each language has its own representation for the sounds and thus its own script, though it was mentioned earlier that some languages may use a common script. In practice, there are small variations in the scripts that probably matter when linguistic aspects are brought in.

Writing Systems

The writing systems for most Indian Languages employ symbols for about sixteen vowels and as many as thirty-five consonants. Syllables that are formed from these basic sounds are also given unique representations. The term conjunct is used to refer to a syllable formed from one or more consonants and a vowel. Though one can theoretically think of thousands of conjuncts, only about 800 of them are known to be in regular use and each of these can combine with a vowel to make nearly 13000 or so individual sounds, each with its own unique representation in the script.

Interestingly, the writing systems employ just about 200 or so symbols to form the unique shapes representing the conjuncts by combining shapes, somewhat in the manner of adding ligatures. For each language, well-defined rules exist for writing most of the conjuncts and their combinations with the vowels. The term Akshara is normally used to refer to a consonant or a vowel or a simple combination of a consonant and a vowel. The term Samyuktakshara is used to refer to conjuncts.

Handling Indian languages on the computer is complicated by the requirement that each and every one of these aksharas or samyuktaksharas be individually recognized. Though only a few hundred primitive shapes may be employed in practice to form the combinations, the large number of aksharas must necessarily be identified individually for linguistic or text processing purposes. Children in India are taught to identify thousands of aksharas and once they have mastered reading the script, they find learning other languages, including European languages, relatively easy.

The methods which work well for a limited set of twenty six different letters in the Roman alphabet, obviously fail or become inadequate when applied to Indian languages, not only for the reason there are thousands of aksharas but also that there is more than one accepted way of writing many of the combinations. Though there are clear rules for writing the combinations, existing practices permit multiple representations for the same conjunct, even within a language, not to speak of variations across the languages.

Codes for the Aksharas

Thus there is need to look at the problem of representing (coding) the large set of aksharas so as to arrive at a standard that can

apply uniformly across all the Indian languages. Electronic text processing can then be attempted using these codes.

The pioneering work, which resulted in the GIST technology at the Center for Development of Advanced Computing, must be regarded as the earliest of the attempts towards some standardization. This development permitted DOS based applications to handle Indian language text. The text was electronically represented using the ISCII code and was largely language independent, thus permitting a uniform approach to dealing with the languages. Over the years, this hardware dependent approach has been replaced by quality word processing and data preparation software but the essential eight bit coding of the characters has been retained. As will be explained later while dealing with Character encoding for Indian languages, eight bit codes are not really suitable for efficient string processing.

All the official languages of the country are written using scripts specific to each language. Scripts denote the writing systems employed by the languages to represent the sounds, which form the phonetic base of the languages. Currently, the following language specific scripts are considered essential.

Devanagari, Gurmukhi, Bengali, Gujarati, Oriya, Telugu, Tamil, Kannada and Malayalam. The scripts for Urdu and Sindhi should also be included in the above, though Devanagari is often used for writing in Sindhi.

The Phonetic Aspect of Indian Languages

The languages of India have a common phonetic base. One does not use the term 'alphabet ' to refer to the set of letters with which the script is written. Instead, the set is called 'Aksharas'. Very Simply, an akshara refers to a sound. Sounds heard in spoken words are built up from the basic set of sounds represented by the vowels and consonants of the language.

In all Indian languages, an akshara is pronounced the same way regardless of its position within a word, unlike in English where the pronunciation varies widely, depending not only on the word but also on the location of the letter within the word.

Also, in Indian languages, the vowels number between thirteen and eighteen while the consonants vary from eighteen in Tamil to as many as thirty-eight in Telugu and Malayalam. All the aksharas are therefore built from about fifty basic letters.

It is indeed possible to use just the vowels and the consonants for writing any of the languages. This is probably how children are taught a script to begin with. In practice however, the scripts abound in what are called 'Samyuktakshars ', which are the equivalent of syllables and represent sounds built up from combinations of consonants and a vowel. The writing system for a language often permits more than one representation (shape) for the samyuktakshar. Samyuktakshars are often referred to as conjunct characters. Clearly, when one sees an akshara in print, its sound is fixed. However, there may be more than one representation for a given conjunct and this depends on the writing practices followed in a region.

All the ideas expressed here may well be grasped by studying the Devanagari script in which Sanskrit and a couple of other Indian Languages are written. An extensive discussion on this can be found in IITM site on Learning Sanskrit (IITM, 2010).

It turns out that when dealing with Indian languages on a computer, one needs a representation for the aksharas in general and not merely the vowels and consonants. The akshara is the basic unit or quantum from a linguistic point of view and computer programs processing text in Indian languages should be able to efficiently deal with this quantum, built up from two or more basic sounds. This poses a real challenge as there

are more than 13000 individual aksharas that have to be reckoned and many more which might come into use, if the need arises.

The importance of Transliteration

The common phonetic base across all the Indian Languages is helpful in situations where language independent information such as statistical data, addresses, schedules of meetings etc., have to be disseminated in different languages simultaneously. People who can speak a language but do not know its script may still be able to read information in that language by merely reading it in a script familiar to them.

Traditionally, books written in English that deal with text in Indian languages such as commentaries on ancient scriptures used Roman transliteration to help read the text. In many instances, diacritical marks were added to the Roman letters to establish a closeness to the aksharas of the language, which would be difficult to achieve with just the twenty-six letters.

Transliteration between Indian languages is very desirable to help people learn one language through another. The common phonetic base makes this easy. Yet, transliteration between the languages will have to be handled with care, for there are quite a few aksharas which are specific to some languages but not seen or used in others. For instance, Tamil does not have the aspirated consonants of Telugu or Sanskrit and reading Sanskrit through Tamil, which is very desirable, is often rendered difficult. Situations such as these are usually handled by introducing new symbols in the script of a language to represent via transliteration, characters found in other languages.

This paper stresses the need to establish a single coding scheme to cover all the different aksharas across all the languages of India in order to allow correct transliteration. In this connection, the use of Roman letters with diacritic marks does result in a script useful for reading text prepared in any of the Indian languages. The National Library at Calcutta has recommended an efficient scheme for Roman transliteration (GOV, NL).

Transliteration Principles

Transliteration refers to the process by which one reads and pronounces the words and sentences of one language using the letters and special symbols of another language. Thus transliteration is meant to preserve the sounds of the syllables in words. Transliteration is helpful in situations where one does not know the script of a language but knows to speak and understand the language nevertheless.

For several decades now, Roman transliteration has been used to represent texts of Indian languages, especially Sanskrit. In many printed books, a key to transliteration would be printed at the beginning in the form of a table. Since it is difficult to represent the aksharas of Sanskrit using just the twenty-six letters of the Roman alphabet, scholars used varying schemes to accommodate sounds that could not be correctly indicated using appropriate Roman letters.

Here are some examples of transliteration as per the schemes, which were in general use in the past. The schemes are somewhat arbitrary in the choice of the Roman letters.

ईश्वरा	Iswara	Eswaraa	eesvara
प्रार्थना	prarthana	praarthanaa	prArthanA
ऋषि	R^shi	ruSi	rishi

Sometimes phonetics symbols are used in place of the normal Roman letters. Phonetic symbols are basically the letters of the Roman alphabet with special marks known as diacritic marks. Here are some examples of transliteration using symbols from the phonetic alphabet. In the second set of aksharas shown below, one sees the use of special symbols from the ascii character set in place of diacritics.

आ	ऊ	ङ	ञ	ळ	ड	श
ā	ū	ṅa	ña	ḷa	ḍa	śa
क	कि	कृ	आर्य		दृष्ट्वा	
ka	kri	kR^	Arya		dR^shTvA	

Roman transliteration which makes use of diacritic marks works better for Indian languages and in the last few decades some standardization has been effected based on the recommendation from the National Library in Calcutta (GOI, NL). Roman letter assignments in this scheme are phonetically equivalent to the aksharas of Sanskrit or other Indian languages. As indicated earlier, the phonetic alphabet with diacritic marks is very helpful for representing text in Indian languages. Such letters are also easily typeset, for typefaces are available specifically for this purpose. Typesetting was however attempted manually for nearly a century until special word processing and typesetting applications were developed using computers. These programs make use of high quality fonts to produce good printouts and displays. However most of them rely on some indirect data entry methods to generate the phonetic symbols.

The primary difficulty in data entry of the phonetic symbols is that there is no provision to input the symbols directly using the standard ASCII keyboard. Desktop publishing and word processing programs provide means by which the glyph code of the symbol is input using the numeric keypad. While this is acceptable, it does not provide a natural approach. Transliteration methods, which use only the displayable ASCII symbols, do not run into this problem since the ASCII letters can be typed in directly. A special computer program would however be required to interpret the input string to produce the Indian languages display or printout. This is precisely what the currently popular transliteration schemes attempt. Schemes such as ITRANS, RIT, ADHAWIN etc., use only the standard displayable ASCII letters and symbols to transliterate the text. These schemes allow multiple representations for certain syllables and long vowels but the processing program handles this well.

Lack of Uniformity Between Different Schemes.

While transliteration based data input is very useful, one must remember that the schemes themselves vary, even for a given language. The consequence of this is that the data entry procedures will change depending on the scheme and worse still, a given transliterated string will produce different outputs for different languages/scripts. Take for instance the word 'yoga'. The transliterated data input for this string using the 'ITRANS ' scheme is 'yogA '. However, when you use this string to get an output in Tamil, using other schemes, you will get ஓயாகா as opposed to யோகா which is the correct transliteration. The fact that the short forms of the vowels 'o' and 'e' are present only in the Southern languages is the real issue here.

Transliteration schemes have to face the problem of letters present in one language and not in the other. Thus, unless a superset of letters from all the Indian Languages is formed, uniform transliteration is ruled out. Even if such a superset were identified, it turns out that unique Roman letter combinations are not easily identified for complex Aksharas. Moreover, the large number of vowels in Indian scripts also add to the complexity in transliteration.

String Processing Using Transliterated Text.

One useful feature of transliterated representation of Indian Language strings is that conventional string processing programs may be used to process the text. However, applications such as sorting will produce erroneous results as the sorting order of the Aksharas and Roman letters are quite different. Many string processing applications such as processing a sentence

may however work properly, so long as the input strings do not contain special characters, which are needed for transliteration but can cause confusion if they happen to be delimiters fixed for parsing routines.

With transliterated input, the representation for syllables is always multibyte with varying number of bytes for different syllables. For example, if we were to examine the aksharas in the second row of the letters seen in the image above, we will see that the last two words contain two aksharas (samyuktaksharas are treated as aksharas since they constitute one syllable) each. However, the word 'Arya' has four ascii letters but the word 'dhR^shTvA' has nine. So linguistically speaking, transliteration using Roman letters may not be the best choice for text processing at the level of a syllable.

It would be helpful to have a representation, which uses a fixed number of bytes for each syllable. Such a representation would be ideally suited for studying the metrical structure of poems or slokas.

Transliteration Features in the IITM Software.

The Multilingual software from IITM has incorporated features to help deal with transliterated text. The multilingual editor has a data entry method that directly allows transliterated text to be typed in and the text viewed in local scripts. A .llf file is also automatically created by the editor. Those familiar with ITRANS based input will find this feature helpful. IITM also has some utilities for viewing and converting transliterated text (IITM).

Data Entry Methods Suited for Indian Languages

Standard QWERTY keyboard seen with most computers is generally used for preparing texts and documents in various Indian languages and scripts.

The answer to the question 'can we do it as simply as one does it for English?' is an obvious NO but a qualified YES. The 'no' part of the answer has to do with the fact that the limited number of keys on the keyboard will certainly not be able to cater to the thousands of aksharas, which occur in our texts. The qualified 'yes' is based on the observation that the keys may be used to represent only the vowels and consonants and thus provide for inputting a series of consonants and vowels from which the required aksharas may be formed using suitable computer programs.

The question of data entry in Indian scripts had attracted the attention of scholars and computer experts since long and today, we can see numerous computer programs, which permit document preparation in different languages and scripts. These programs, some of them very good in many respects, tend to differ significantly in their approaches to data entry. The variety seen in their approaches merits discussion so that we may better understand the problems involved.

The programs permitting data entry in many Indian languages/scripts may be classified based on the specific approach taken to forming the aksharas from the keystrokes. These are listed below.

- Language/script specific data entry , which relies on a specific font.
- Transliteration based data entry.
- Data entry conforming to Manual Typewriter keyboards, specific to each language.
- Data Entry based on the INSCRIPT layout
- Data entry methods specific to generating HTML pages supporting our scripts (web page creation)
- Data entry based on uniform mapping of the keys for all the languages/scripts.

Data Entry Methods Which are Based on Fonts.

The font based data entry methods utilize the feature supported by conventional word

processors where the font to be used for displaying the text may be dynamically selected/changed during data entry. Today, the font rendering capabilities built into the operating systems are quite sophisticated in that the required shape of a character may be built from several primitive shapes which are called glyphs. Each font may consists of about 200 different glyphs, where each glyph may directly represent a letter of the alphabet, a special character or a symbol.

In fonts for Indian languages, the glyphs will invariably include shapes for the matras, special samyuktakshars and special ligatures besides the basic shapes for the vowels and the consonants themselves. When a font is selected, the word processor will display the glyphs corresponding to the keys entered. For the English language (Roman alphabet) each letter corresponds to only one glyph in the font and data entry is smooth. In the case of Indian scripts, we will have to know what keys will have to be entered to display the sequence of glyphs, which will make up one character. In the case of Roman, the set of displayable glyphs correspond to the set of ASCII codes that are generated when keys are pressed on the keyboard. This is a set of 96 characters and anything more than this number will require special data entry, as keyboard has only a limited set of keys.

Conventional word processors are designed for languages where a letter (or a character) to be displayed is associated with one glyph only. Also, for most of the western languages, the character set itself is limited and so the set of displayable characters is well within the 96 mentioned above. Even though a font for western languages may need to accommodate only the displayable set, many glyphs in the font may be present that are not displayed when the keys are pressed during regular data entry. That is, there may be glyphs in a font, which are displayable but not necessarily shown when keys are entered. These glyphs usually correspond to characters with accent marks, specialized symbols, diacritic marks etc., and may be required mostly in printed text and special applications. Some word processors do support data entry for these glyphs, which are typically located in the upper ASCII range (160-255) by allowing the numeric value of the glyph location to be input with the ALT key kept down as the numeric value is typed in.

Fonts for Indian languages (Except Tamil) are required to have many more than 96 glyphs and so, data entry based on this method of inputting the numeric glyph code values and displaying the character, will become necessarily cumbersome. Worse still, the input sequences are font specific and will vary from font to font even for a given script. Fonts for Indian languages had till recently evolved arbitrarily and do not follow any standard. Consequently one sees wide variations in the glyphs themselves as well as the encoding for the font which used to locate the glyphs at specified locations in the table of 256 locations, there are no standards for glyph locations for Indian scripts and it considered that such standards will be difficult.

A discussion of fonts and the issues to be considered in designing Indian language fonts will occur in later sections.

A point to keep in mind is that the internal representation of the text prepared according to this method is in the form of eight bit glyph codes. This has serious consequences if one were to attempt any sort of string processing of the text because the glyph codes bear no relationship whatsoever to the linguistic nature of the aksharas in terms of lexical ordering, sorting or indexing etc. Yet, this font based data entry method is popular with DTP packages, where one is interested more in printing text as opposed to linguistic analysis.

There is however a bright side to this approach. Though keyboard entry is cumbersome, one might effectively use the

cut and paste facilities supported in the word processors to perform some editing of the entered text. In some word processors, one also sees an image of the keyboard with aksharas and matras assigned to the keys and the user may simply click on the keys to select the glyph to be displayed. Also if the user were to keep a standard file containing the glyphs, then individual glyphs may be cut and pasted even for entering short sentences of text. Some Urdu word processors have this feature.

The data entry on the basis of fonts and glyph codes cannot really provide a natural interface, even if supported through sophisticated macro facilities found in some word processors. When we have to input a text like the following multilingual text using our favorite word processor or DTP program there is a need for some easy ways to do this.

Here are some conjuncts in different scripts

क क्र ग्र द्य ट्ट त्र ह्म द्ध द्भ म्ल प्र क्ष स्त्र

ಕ್ಕ ಜ್ಜ ಗ್ರ ದ್ಯ ಷ್ಟ ಶ್ರ ಹ್ಮ ಧ್ಯ ಭ್ರ ಮ್ಲ ಪ್ರ ಕ್ಷ ಸ್ತ್ರ

Transliteration Based Data Entry Methods.

Transliteration has been a popular approach to preparing printed documents in different Indian scripts. The idea behind the method is to use Roman letters to represent the aksharas of the languages and process the resulting string (ASCII text) using special computer programs, to produce printed output. The output is obtained using appropriate fonts.

One of the early computer programs to successfully implement this idea is the Dvng processor for Devanagari using TeX. This program produced a TeX file, which could be typeset using the TeX program. Franz Velthuis who devised this package, had also included a special Devanagari font for use with the package. The Dvng package ran on Unix systems and TeX fonts have the advantage that nearly every glyph in the font, which may have as many as 250 glyphs, may be used in printing. In contrast, fonts for other systems such as X-windows, MSWindows, PostScript etc., are restricted to just about 200 glyphs. This is not a design limitation of the font but a problem arising out of the inability of application programs and font rendering routines to look at specific glyph locations. As a consequence of the rich set of glyphs, the Dvng package could print a rich set of conjunct characters in Devanagari.

After Dvng, Charles Wikner enhanced the fonts to accommodate Vedic symbols and also gave a new processing package. As of today, the Devanagari output obtained using this package is of remarkably high quality and Wikner's choice or design of the glyphs has allowed nearly a thousand different conjunct formations to be derived from the basic set of about 250 glyphs.

Both the packages mentioned here had arrived at some guidelines for standardization in the selection of the Roman letters for the aksharas of Sanskrit. In many instances, special symbols from the ASCII set were required to be used to distinguish similar sounding aksharas. Printout using these packages were restricted to Devanagari but Roman could be part of the text as well, permitting bilingual outputs. Subsequently TeX based systems were introduced for Tamil, Telugu, Malayalam, Gurmukhi, Gujarati and Bengali.

Following the success of the TeX based packages, Avinash Chopde developed a special transliteration package that allowed other scripts to be handled as well, via language specific fonts. His ITRANS package is well known on the web. Subsequently he enhanced the package to work with normal fonts under Windows-95 and X-windows and was able to generate html documents for display on the web. The most recent version of ITRANS supports quite a few languages.

It must be remembered that all transliteration based data entry methods, require a computer program to generate as well as format the output and hence they cannot be applied or used for interactive data preparation, where the display in Indian scripts immediately follows the key strokes.

The ITRANS package was followed by JTRANS, a Javascript based program by Sandip Sibal who allowed quick generation of html documents from transliterated inputs. This package introduced Xdvng, a quality font for Devanagari, which could be used for viewing web pages with Devanagari text both under MSWindows and XWindows. Sibal's package is however restricted to Devanagari.

The Itranslator package from Onkarnath Ashram in Rishikesh allows data entry in ASCII using the ITRANS scheme but allows the string to be converted to Devanagari and displayed on the screen itself. The font used by this package is probably the finest of the freely available fonts for Devanagari and is known as Sanskrit_1.2. Unfortunately, this font is suited for the Windows platform alone and has glyphs in locations that create problems on other platforms. A later version of this font (Sanskrit-98) seems to avoid the above problem. There are many later additions to the Itranslator package, including new fonts.

Transliteration Schemes for Tamil and Telugu.

There have been a few popular packages for Tamil and Telugu, which use the transliteration, based data entry method. The Adhami package was written for use under DOS and subsequently enhanced to work under MSWindows and produced displays and printouts in Tamil. Other transliteration schemes such as Mylai and Cologne were also popular with Tamil. For Telugu, the RIT package developed by Rama Rao Kanneganti, used TeX for typesetting the output. Details of some of the transliteration schemes may be found elsewhere in this study.

Universal Transliteration Scheme for Indic Scripts.

Dr. Anthony P. Stone has recommended a special transliteration scheme to handle all the Indian scripts. This interesting proposal uses eight bit character codes to represent the vowels and consonants and hence maps a fairly large superset of vowels and consonants from all the scripts of interest. This is a meaningful proposal but has only one likely limitation. Existing data entry facilities do not permit easy typing of characters in the upper ascii range (160-255) and so data entry using this scheme was not feasible. However, it is quite easy to display all aksharas using this scheme. Therefore printouts of Indian language texts in transliterated form, can be easily generated. Standardization of transliteration will help considerably in dealing with Indian languages in a uniform manner.

Summary of Transliteration Based Data Entry.

1. This method allows text in Indian languages to be input using Roman letters. A special computer program is used to process this text in Roman to produce printouts or displays using appropriate fonts for the scripts. There are several transliteration schemes in use. Most of the processing programs run under Unix.

2. Transliteration schemes are often specific to one Indian language/script. There is no single scheme yet that correctly handles all the Indian languages.

3. Phonetically close Roman letters may not be found for all aksharas. So some compromise is required in selecting the Roman letters. Also multiple representations for the same akshara seem to be allowed, making the processing somewhat complex.

4. It is possible to confuse most of the processing programs by inputting arbitrary formations of conjunct aksharas.

Transliteration based data entry is a workable solution for Indian scripts, since in principle,

it allows for a uniform data entry mechanism for all the languages. The transliteration scheme should be comprehensive enough to handle all the aksharas across all the languages/scripts.

Will it be meaningful to have a system where, as one types in the transliterated text, the actual characters of the Indian script appear on the screen? This is what was being attempted by some of the applications, which work under Microsoft Windows systems. While this was an interesting development, the transliteration schemes used are often language specific and have not always permitted the formation of many complex conjuncts (Samyuktakshars).

Manual Typewriter Keyboard Based Data Entry.

Manual typewriters for different Indian languages have been available for quite some time and their use in Educational institutions and Government offices is substantial. Manual typewriters provide for a minimal set of aksharas consisting of the basic vowels and consonants together with the matras so that text can be prepared conforming to the writing system for the language. The location of the keys for the vowels and consonants on a regional language typewriter is specific to the language. Many are adept at using such typewriters and when they have to move over to using word processors, they would rather see the same keyboard mappings. Some word processors do provide for data entry based on the typewriter based key mappings. The resulting text may not include a number of conjuncts but will be entirely adequate for normal modern day correspondence.

Data Entry Based on the INSCRIPT Keyboard.

The INSCRIPT keyboard allows more or less uniform data entry of text across the different scripts. The mapping provides for the data entry of vowels, consonants and matras consistent with the specifications in ISCII. The INSCRIPT layout utilizes only the keys provided on a standard QWERTY keyboard and is hence implemented easily on personal computers. It may be observed that a number of keys normally used for punctuation or special symbols are also mapped to the ISCII characters. It will therefore be difficult to perform data entry of text along with a full complement of punctuation marks, which have come to into use with almost all the scripts. Microsoft applications also use the INSCRIPT layout for Unicode data entry and hence suffer from this problem. The Microsoft Hindi keyboard has apparently provided for many punctuation marks but one has to effect multiple keystrokes to enter them. Shown below is the INSCRIPT layout on a QWERTY keyboard. Keys corresponding to the ISCII characters are common across all the scripts.

Special Programs for Web Page Creation.

During the past several years, display of Indian language text on the Internet Newspapers and Magazines has become popular. Web pages in Indian scripts are feasible on account of the fact that web browsers may be asked to display a given text in a specified font. This will be further discussed later.

The html standard provides for an interesting way of specifying the glyphs to be displayed either through the numeric code assigned to the glyph or the universal name assigned to that glyph location consistent with the font encoding that has now become standard. This way, the html language also functions as a macro language, where a text string describing the glyphs to be shown may be just typed in using standard ascii. While one may not need to worry about this for glyphs in the displayable ascii range, the approach is very useful for glyphs in the upper ascii range. In lighter vein, some people on the net refer to this as the method for the 'ASCII impaired'! The advantage of this approach need not be emphasized, for virtually any text editor capable of data entry for the upper

ASCII characters can be used to produce web pages in Indian languages, provided one has patience!

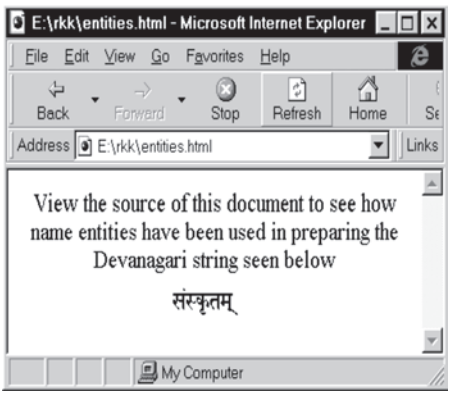
As an example, the html document shown below will produce the display given in the image that follows. < and « represent two glyphs that are specified through their name entities.

<html>

 View the source of this document to see how name entities have been used in preparing the Devanagari string seen below

s<Sk«tm!

</html>



The user preparing the html document must necessarily know the location of the glyphs. This, as we know is font specific, even if the font is meant for a specific script.

In a sense, generating display through html is similar to the macro-based approach taken by TeX, the typesetting program developed by Dr. Knuth. While TeX has the advantage of using most of the 256 glyphs in a font, html displays are constrained to using only about 200, thus loosing the ability to display some conjunct letters.

Web Pages Supporting Display of Unicode Text.

Unicode has been accepted as a meaningful standard for handling multilingual text. Most browsers introduced after 2002, include support for this. With Unicode text, the method indicated above does not apply, for the encoding standard automatically identifies the font to be used. Unfortunately, rendering Unicode text in Indian languages is beset with multitudes of problems and it is unlikely that correct rendering of text will be realized. Unicode text in Indian scripts will have to be created using appropriate programs such as Microsoft Word and related applications. Even in 2005, several browsers cannot correctly display Indian language text represented through Unicode. The difficulties encountered in dealing with Unicode for Indian languages will be explained in greater detail in later papers of this series.

Phonetic Mapping of the Vowels and Consonants.

One way of looking at data entry in Indian languages is to view the text as consisting of aksharas that can always be decomposed into vowels and consonants and perhaps a few

symbols. In this phonetic approach to data entry, just one keystroke is associated with each vowel and consonant and a computer program typically an input module in an application, keeps track of the keystrokes and forms the aksharas. In many ways, this approach is similar to the transliteration based data entry except that we are not constrained to mapping the vowels and consonants to any specific keys. Also, in the transliterated input case, more than one keystroke may be required to form a vowel or a consonant (e.g., an aspirated consonant or a diphthong).

The Inscript keyboard layout; the recommended standard for ISCII based systems follows this approach though it includes keystrokes for the matras as well. Since the addition of a matra to form a consonant vowel combination is not uniformly applicable to all cases (in Tamil and Malayalam, the combination with the vowel 'u ' changes the shape of the consonant), the Inscript keyboard does not correctly indicate or reflect what would happen when a combination is input. However it may be assumed that the key for a matra does not always result in a matra but may change the shape of the consonant. The inscript keyboard basically confirms that a phonetic approach to data entry is feasible. True, the basic requirement here is that the input module must process each keystroke taking into consideration the previously entered keys and also check if the conjunct is valid or meaningful. But this is a module that can be written once and incorporated into an application program, to work uniformly across all the Indian languages.

The data entry scheme recommended for the IITM software essentially follows this approach with one additional facility. The CTRL key or an equivalent is used to indicate that a combination is required to be effected with the previously formed akshara and the current input. Thus the user explicitly indicates that a conjunct will have to be formed. This feature is helpful in situations where consonants and vowels not present in a language are attempted to be input. The system will not accept such inputs thus providing a safeguard that only valid combinations may be input.

In the phonetic approach, the key mappings do not relate to the generic consonants i.e., a consonant without any vowel. The mapping relates to the form of the consonant where the first vowel 'ah ' is assumed to be present. This is often the way the consonants are taught for children. This way, only one keystroke will be required to enter the most frequently required form of each consonant, as opposed to the case with transliteration based data entry where two keystrokes will be needed.

References

Dash, N S (2005). Corpus linguistics and language technology: With reference to Indian languages. Mittal Publications.

Pal, U., & Chaudhuri, B B (2004). Indian script character recognition: a survey. Pattern Recognition, 37(9), 1887-1899.

Shneiderman, B (2010). Designing the user interface: strategies for effective human-computer interaction. Pearson Education India.

Singh, A. K. (2006, October). A computational phonetic model for Indian language scripts. In Constraints on Spelling Changes: Fifth International Workshop on Writing Systems.

Sinha, R M K (1984). Computer Processing of Indian Languages and Scripts—Potentialities & Problems. IETE Journal of Research, 30(6), 133-149.

Sinha, R. M. K. (2009). A journey from Indian scripts processing to Indian language processing. IEEE Annals of the History of Computing, 31(1), 8-31.

Sinha, R. M. K. (2009). A journey from Indian scripts processing to Indian language processing. IEEE Annals of the History of Computing, 31(1), 8-31.