Multilingual Computing for Indian Languages - An Overview

Shivashankar B Nair

Abstract

While English has predominantly maintained its lead both as the lingua franca of the Internet and also for basic man-machine interactions, the need of the day is a system that can cater to the native by facilitating such communication in the language he is most comfortable. This calls for the realization of Multilingual Systems that can present the same information in a variety of languages. The term Multilingual Computing in the present context refers to systems that are capable of running programs that accept, process and present data in more than one natural language. The concerned language for interacting with the computer may be selected at the time of invocation or use of the program. Such multilingual systems do not fully overcome the man machine barrier. The user still has to comprehend and react to the cryptic messages presented in the language of his choice. The problem can be overcome only by the use of natural language processing systems that can translate user commands and queries in his native language to machine level commands, facilitate execution and present the results in a manner that is very akin to the responses of a human being. This paper discusses the basic issues in the formulation of such multilingual systems.

Keywords : Indian Language, Multilingual Computing, Natural Language Processing.

0. Introduction

The advent of the Internet and the concept of ubiquitous computing have lead to a plethora of applications of computing systems. Information is now available virtually at every point but in the form of web pages or some such related form via the Internet. Web sites that host information in a wide variety of areas ranging from commercial, scientific, medical to educational and literary aspects are currently available. A person literate in computers does not really feel the pinch of accessing and weaning out the information he desires. On the contrary, a person not well versed in the same faces numerous diverse problems. Since most of these sites host information in English, those comfortable in this language find accessing information a lot easier. The net effect is that most people who wish to use computers have to essentially possess good knowledge of English. Though many countries have now opted for using operating systems in their national languages, accessing information across the World Wide Web still requires knowledge of other languages. The problem is compounded in countries where people speak more than one language. India is possibly one of the best-known examples. With about twenty two official languages, and many more unofficial ones being spoken prominently by large populations, this country faces an uphill task of disseminating information not just in English or its national language (Hindi) but also in a diverse set of languages that are grossly different in their phonetic and linguistic aspects. Keeping this problem in mind the Ministry of Information and Communications Technology of the Government of India [1] took up an initiative in the year 2000 to set up several Resource Centres for Indian Language Technology development across the country, to act as language clinics. Assigned with one or more Indian languages, each Centre aims at developing tools for multilingual computing. These include editors, morphological analyzers, spell checkers, language corpora and the like, all essential aids for a proper multilingual computing and language processing environments.

The recent information technology boom and the widespread use of computers have still not percolated to the common man. One of the prime issues that contributes to this problem is the absence of a user-friendly human-computer interface. The advent of graphical user-interfaces has, to a great extent, made the

use of computers an easy task. But these interfaces have their own disadvantages. They provide near pictographic information of what can be done but provide no more. A user may want to do more than just click on icons. This calls for a higher degree of sophistication - systems that can accept commands in speech and also talk back. Realizing such systems falls under the domain of speech and natural language processing, both fairly simple tasks for human beings but highly complex ones for computer systems. This paper describes the meaning of multilingual computing, the techniques used to realize it and winds up with an overview of language processing and the tools required in such a scenario. The latter being a vast and only partially solved problem domain is only described in brief.

1. Multilingual Computing

As the very term suggests, multilingual computing refers to using the computer to communicate with humans in different languages [2]. The term also encompasses browsing and searching the World Wide Web in different languages, even typing from right to left as in some languages like Arabic and printing on different paper sizes. Issues like transliteration, spellchecking [3] and reading in one language and typing and printing in another also fall in this domain. Multilingual computing also deals with mechanisms for accepting input in the desired language, rendering and storing information. It is not only a software and a hardware problem but also one that deals with conforming to world-wide standards [4]. This is because each language may otherwise use a different mechanism for input, rendering and storage that can lead to problems in compatibility.

2. Representation of Information

Information is represented in the form of codes, the most common being the ASCII (American Standard Code for Information Interchange). ASCII is a 7-bit code that encodes 32 control characters, plus 96 alphanumeric characters (A to Z, 0 to 9 plus symbols). The Extended ASCII is an 8-bit code and facilitates more information representation.

2.1 ISCII

Many Indian languages use the ISCII (Indian Script Code for Information Interchange) code [5] that allows usage of 7 or 8 bit characters. In the 8 bit mode, the lower 7 bits (128 combinations) comprise the ASCII character set while the upper 128 characters cater to the Indian scripts. In the 7-bit mode the control character SI is used for activating the ISCII code set. A script is a set of symbols required to represent a writing system. It can be used to represent many languages. For instance the Latin script is used to represent languages like English, French and German. Most Indian languages have evolved from the Brahmi script. They have a common phonetic structure making a common character set viable.

ISCII has been christened lately as ACII (Alphabetic Code for Information Interchange) and now caters to scripts of SAARC countries. ACII is an 8-bit code that has the ASCII character set on the bottom half while the upper half is occupied by ACII characters. In the Indian context, ACII accommodates around 10 Indian scripts including Assamese, Devnagiri, Malayalam and Punjabi. The basic characters have been positioned to enable direct sorting.

It is thus obvious that 8-bit character codes are adequate for languages which have a small alphabet set and when written text in these languages comprise of individual alphabet and punctuations. Things however become difficult for languages, which do not satisfy these conditions. This is true for Indian languages, which have conjunct characters that express combinations of sounds. Data entry thus becomes a problem in such cases. Before displaying the final version of the character (or conjunct), the terminating vowel has to be determined so as to generate the shape to be rendered. Thus an algorithm to tackle variable number

of bytes, comprehend and translate them to a shape formed using one or more glyphs is required. In case of Roman letters such a problem does not arise as each byte corresponds to a unique glyph.

To make things more vivid we take a look at a simple example.

Input String System Alphabet followed by the respective Codes

trupti ASCII 116(t) 114(r) 117(u) 112 (p) 116(t) 105 (i)

ISCII 264(cT) 258(0)184(^)227(0) 176(*T) 258(O) 264(cl) 223(6t)

It can be seen that the characters for the ISCII code by themselves carry no meaning. A program has to find the sequence of codes entered, comprehend the order and finally aggregate the associated glyphs and send them to the rendering device. The string however may be stored as a chain of associated ISCII codes.

2.2 Unicode

Of late the problem of multilingual computing has become an international issue. More number of scripts and languages has compounded the problem. New attempts were therefore made for standardization of multilingual documents. A new code to support all languages was looked into. A new code termed the Unicode was evolved in the year 1991. Unicode is an extension to the ASCII code but also accommodates International languages and scripts. Most international languages require only 7 or 8-bit character codes, as they need only a small set of symbols to represent the letters. Unicode permits the use of this 8-bit representation but augments the code with an extra 8-bit language identifier. These extra 8 bits that form the most significant byte of the 16-bit Unicode can cater to 256 different languages and about 128 characters per language. Thus Unicode compatible software can identify the language of each character and also use the appropriate rendering system. For Indian languages Unicode has greatly conformed to ISCII. Minor variations however do exist for some languages. The embedding of ISCII has resulted in the fact that Unicode inherently bears with it all drawbacks posed by ISCII. A more comprehensive discussion on the limitations of Unicode and ISCII can be read from [6].

3. The Concept of Fonts

The previous paragraphs discussed how text is represented in a coded form and interpreted for rendering. Fonts provide for displaying the symbols on the rendering device. From the computer's point of view, a font is a file (or files) required to display and print in a particular style. A set of fonts that are similar in looks but have different attributes is termed as a font family. Fonts may be of various types. A bitmapped font, at times referred to as a screen font, contains all information regarding the pixels. This information is used by the rendering software to display the font on the screen. Printer fonts generally referred to as PostScript fonts comprise of more than one file. These files include the bit mapped screen font files and a printer font. The former are used for display and the latter by a PostScript compatible printer. The advantage is that printouts are smooth unlike those made using bitmapped fonts. TrueType fonts allow smooth screen displays and facilitate printing without the use of extra screen font sizes or PostScript. They can print on non-PostScript printers. Another category called Dynamic fonts allows delivery of TrueType fonts to the client side.

Fonts are generally stored in an OS specified directory called the font folder. At the time of booting the system checks for these fonts and activates them. Alternatively a font manager may be used to activate or deactivate fonts. When a key is pressed, the keyboard driver looks into the code generated by the keystroke, interprets the same and passes it onto the rendering software. This in turn refers to the font files, grabs the information about the associated glyph and renders it on the screen. As mentioned earlier rendering

conjuncts requires an extra element of processing to aid the exact rendering on the screen. A point that may be emphasized here is that keyboard layouts may also vary. A Romanized keyboard layout uses phonetic English mappings to compose the text. For example, the keystrokes *aa* or A would give the *matra* for the corresponding language. The Typewriter Layout on the other hand is structured based on the normal Hindi Typewriter layout. This enables typists to easily adapt to the keying system. The Phonetic layout, standardized by the Department of Electronics of the Government of India, has the same layout for all Indian Languages. This also facilitates transliteration and ease of typing across languages.

The commonest way of displaying Indian language text in Web pages is the HTML based approach. The problem here is that some browsers may not support the encoding of the specified font. It may happen that the user has to manually switch to the correct encoding. Another approach comprises of using dynamic fonts wherein the fonts specified in the pages are sent along with it. However there are restrictions again in the type of encoding used. They generally work only with true type fonts and may not be rendered properly in Unix systems.

4. Transliteration

This is the processes of converting each alphabet in one language to the equivalent in another. Transliteration is advantageous in the Indian language context due to the common phonetic structure possessed by these languages. Unlike in English, the *aksharas* (mildly equivalent to the alphabet) in Indian languages, which refer to sounds, are pronounced the same way irrespective of their position in the word. This property facilitates easy conversion from one Indian language to the other. Thus if a proper noun or an address or date were written in an Indian language which the user does not comprehend, he can easily transliterate and read the same in a language he is well versed with. For instance a title of a book written in Marathi can be read in Assamese. This also helps people learn a new language.

5. Natural Language Processing

Natural Language is the language used for communication amongst human beings in the real world. The term *real world,* makes the problem much more difficult. If we were to converse using telegraphic language, building systems that understand such conversations would become a much simpler task. For instance consider understanding the two sentences -

1. We all should use telegraphic language.
2. Use telegraphic language.

It is apparent that the second sentence is more precise and extracting its semantic content is obviously easier and quicker. It is thus obvious that the main aim of natural language understanding systems is to translate natural language sentences (of type 1) into a comparatively simpler form (of type 2). Type 2 sentences are finally used by systems aided by some *a priori* or gained logic to comprehend and execute desired actions.

While the term Natural Language (NL) refers to the language spoken by human beings, Natural Language Processing (NLP) refers to an area of Artificial Intelligence (AI) that deals with systems and programs that can accept, comprehend and communicate in natural language. Systems that are capable of processing and understanding natural language bridge the man-machine communication barriers to a great extent. They facilitate interaction with computers without resorting to the memorizing of complex commands. Computational Linguistics (CL) is a closely related and often talked of field in conjunction with Natural language processing. It uses knowledge of both Linguistics and Computer Science to study the

computational aspects of the human language faculty. There is thus a major overlap in the areas AI, CL and also Cognitive Science.

While speech recognition systems convert the speech input to a textual representation, natural language processing systems perform the job of understanding the meaning of the input. NLP entails several phases. As part of the initial processing stage, a morphological analyzer does the job of finding the root words of each word or token within the sentence. These root words along with the associated morphemes allow us to understand the correctness of the word and also the inflections. This phase also facilitates spell checking and correction. The second phase of processing is carried out by a syntactic analyzer, often referred to as a parser. It verifies the grammatical correctness of the sentence and finds whether or not the various words conform to the grammar of the language. Several types of grammar representations and associated parsers exist [7,8], the more common one being the context free grammar. The grammatical representation thus generated is then used by the semantic analyzer to wean the actual meaning of the sentence. The analyzer uses semantic and pragmatic knowledge in the extraction. In most cases this is a very domain specific job.

6. Multilingual Computing and Natural Language Processing

Though definitions of Natural Language Processing do not really include Multilingual Computing, experience has shown that realizing NLP systems across languages requires more than just understanding the linguistic, morphological, grammatical and semantic aspects of the language. From the Engineering perspective, it requires everything available in the domain of multilingual computing. From the perspective of Multilingual Computing too the understanding of many language-related issues such as phonetics, linguistics, etc. are highly necessary to fix and standardize the manner in which the system is to work and deliver. The point being emphasized here is that the two areas are greatly interlinked and issues in one significantly affect the other. Efficient machine translation systems can be formulated but their viability and scalability depend on the quality of the multilingual computing platform.

Almost all linguistic theories and mathematical and computational models are formulated so that they are in principle applicable to natural languages at large, and not just to one or more specific languages. Computational models that deal with more than one language form the field of multilingual NLP. Machine translation (MT) is a sub-category of this field. Conventional MT systems try to produce one translation for the source sentence, but, of course, in order to achieve this, as is well known, all kinds of other information are usually needed. This includes discourse context (i.e. the previous text), the context of the situation (particularly if it is a dialog), and a variety of domain-knowledge, exactly the kind of information that a natural language understanding and generation system needs. Though a clear-cut solution to real MT is definitely not around the corner, there are systems constrained to specific domains that have been attempted and even put to use. Generation of multilingual texts from a source text that is prepared in a greatly constrained, unambiguous, highly stylized language for instance is one such area. These systems form tools to translate manuals written in one language to several others. There are others like the TAUM METEO system that translates weather reports from French to English. The system has been in operation for quite some years in Canada. In weather reports the sentences are usually short, highly stylized and mostly employ a standard phraseology. SYSTRAN is another effort to translate between several European languages.

In India too there have been serious attempts at machine translation. Anglabharati [9] is one such English to Hindi translation system currently available on the web. The system uses the *Interlingua* technique for translation and is thus scalable. The English sentence is first translated into an *Interlingua* termed as PLIL (Pseudo Lingua for Indian Languages). A text generator then transforms it into its equivalent translation in the target language, Hindi in the present case. Thus if text generators for other languages are written the same PLIL can be converted into the language of choice facilitating one to many language conversion.

Attempts are currently being made by many Resource Centres across India to write such text generators for the assigned languages.

7. Tools for Multilingual Language Processing

Multilingual Language Processing calls for the existence of several tools required at various stages of processing. One or more tools may be instantiated at each phase of processing. Some of the more important ones have been briefly described below.

i. Corpora

A *corpus* is a collection of a large number of pieces of a language stored in a standard form based on some explicit linguistic criteria. They are used to serve as a sample of the language. Several types of corpora exist. A *tagged corpus* contains words with their parts of speech alongside. *A. parallel corpus* contains a collection of texts translated into one or more languages other than the original. In its simplest form it comprises of two languages with one of the corpora containing the exact translation of the other. Such corpora are useful in translation from one language to the other.

ii. Morphological Analyzers

These analyzers are required to find the stem or the root of a word. By doing so a word is split into its basic parts called morphemes. Such analyzers require a high amount of linguistic information to be embedded. Thus construction of such analyzers entails collection and representation of knowledge provided by expert linguists in the concerned language. The process aids in finding whether the word is a valid one in the language.

iii. Spell Checkers

Spell Checkers form a vital tool and perform the job of detecting errors in the text and suggest the relevant corrections. They use a lexicon or a dictionary of words or even the corpus and together with the morphological analyzer perform the jobs of detection and correction.

iv. Parsers

These enable checking the grammar of the sentence in question. Panian grammar used for Indian languages has been described in [10].

v. Multilingual Dictionaries

These dictionaries provide for machine translation by providing the equivalent words, category, etc. in the target language. Since no dictionary representation standards have evolved so far the formats in which data is stored have to be known a priori.

8. Conclusion

The mythological problem caused by the Tower of Babel culminating in the Confusion of Tongues has found its way into the era of computing. Large amounts of information in numerous languages have forced us to think seriously of standardizing methodologies for information representation. The birth of Unicode may pave the way to a better future for greater flexibility in the use of multilingual computing platforms. Natural language processing, a vastly incomplete area of research, may reap the benefits of such systems.

9. Acknowledgement

The author wishes to express his gratitude for the support received from the Resource Centre for Indian Language Technology Solutions at the Indian Institute of Technology (a project funded by the Ministry of Information & Communication Technology of the Government of India), as also Samir Borgohain and Monisha Das, Project Personnel at the Centre, in the making of this paper.

10. References

1. http://tdil.mit.gov.in
2. http://LanguageLab.csumb.edu
3. Kukich, K., Techniques for Automatically Correcting Words in Text, ACM Computing Surveys, Vol.24, No.4, December 1992, 377-439.
4. Goodwin-Jones, R.; "Emerging Technologies Language & Learning", Language Learning and Technology", Vol.6, No.2, May 2003, pp.6-11.
5. Viswabharat@tdil, Language Technology Flash, Jan 2002.
6. http://acharya.iitm.ac.in/multi_sys/uni_iscii.html
7. www.link.cs.cmu.edu/link/
8. Joshi, A.K., "Natural language processing", Science, Vol.253, No. 5025, September 1991, pp. 1242-1249.
9. Sinha, R.M.K.; "Machine translation : an Indian perspective", Proceedings of the Language Engineering Conference. LEG 2002, 2003, IEEE Computer Society, Los Alamitos, CA, USA, pp. 181-182.
10. Bharati, A.; Chaitanya, V.; Sangal, R.; "Natural language Processing - A Paninian Perspective", Prentice hall of India, 1995.

About Authors

Dr. Shivashankar B Nair is Associate Professor in Department of Computer Science and Engineering at Indian Institute of Technology, Guwahati, Assam.
E-mail : sbnair@iitg.ernet.in