Issues of Document Engineering in the Indian Context

Atul Negi, Kavi Narayana Murthy
Department of Computer and Information Sciences,
University of Hyderabad, INDIA
email: {atulcs,knmcs}@uohyd.ernet.in

Abstract

In this paper[1] we discuss the core issues relating to document processing of Indian language documents. We emphasize the distinction between the entities *language, script, and font*. *Script grammar*, unique to Indian languages, is used to define the concept of characters in India scripts. Issues of encoding, processing, rendering and recognition are presented in the context of script grammars. A deep understanding of these concepts facilitates good system architecture and design of text processing systems, OCR systems, web documents etc. We give an example of an OCR system developed for Telugu, a major Indian language. The OCR system is structured with a core OCR engine which recognizes basic shapes and a separate module that composes script level syllables for a character encoding standard such as ISCII or UNICODE.

Keywords: Script Grammar, OCR

1 Introduction

Document engineering in a multi-lingual environment is becoming increasingly important, both due to internationalization and the need for local language support in multi-lingual countries like India. Indian languages and scripts are characterized by certain unique aspects, differing significantly from other language families of the world. These issues have not been very well understood in the document engineering discipline. In this paper we sketch the relevant features of Indian languages and scripts and show their implications for document engineering. Text representation and processing, structure of web documents and issues relating to Optical Character Recognition (OCR) are taken up. Examples are included from the *DRISHTI* OCR system developed by us for Telugu and other Indian languages. However, the issues raised here are generally applicable to all Indian languages and scripts.

2 Indian Languages and Scripts

India is a country of one Billion people, about one sixth of the whole world population. India is also an ancient civilization, dating back

[1] The research reported in this paper was supported by the University Grants Commission under the research project entitled "Language Engineering Research" under the UPE scheme

to at least four thousand years. Contrary to common belief, there are as many as about 150 different languages spoken in India today. These are not dialects - dialects add up to a much larger number. Many of these 150 or so languages have not yet been studied in any great detail. Of these, 18 major languages have been given constitutional recognition. These major languages include the official languages of the federal states of India and are among the most widely spoken languages of the world. These languages span four different language families - the Indo-Aryan, the Dravidian, the Tibeto-Burman and the Austro-Asiatic families.

The scope of this paper is restricted to the major languages and in particular to Telugu, a Dravidian language spoken mainly in the southern state of Andhra Pradesh. Telugu is the second largest spoken language in the country and is also one of the most complex. Our focus here will be on issues relating to the Telugu script.

2.1 Indian Scripts

The difference between language and script is sometimes missed out, especially when there is a one to one correspondence between Languages and Scripts. In fact language *is* speech and writing is only an artifact. A language can exist even without a script, there *are* languages without script. There are people who 'know' their language very well, are quite proficient in the use of language, and yet do not know reading or writing. All of us learned to speak first and learned reading and writing only much later. Learning to read and write requires training and conscious effort. *Language and script are two different things.*

It is also conceivable that a given language is written in several different scripts. This is indeed a fact as far as Indian languages are concerned. Indian scripts have existed for, and evolved over, thousands of years. There are in fact many scripts which are in use today. Being a vast country that has always encouraged plurality in every aspect of life, it is not surprising that there are so many scripts [4]. More importantly, the correspondence between languages and scripts is not a strict one to one. *A language may be written in many scripts and a script may be used for writing several languages.* Thus Sanskrit is written in not only the Devanagari script but also in almost all other scripts. Devanagari script is also used for Hindi, Marathi and Sindhi languages. Telugu is mostly written in the Telugu script.

There is a tradition of mixing different languages and scripts in the same document. Many government documents are required to include English and Hindi and a third local language may also be used. Currency notes are multi-lingual. Sign boards are multi-lingual. There is a tradition of mixing Sanskrit with local languages and Sanskrit then gets written in the script used for the local language. Mixing English with local languages is also very common and English words are often transcribed in local scripts. Thus you may frequently encounter multi-lingual texts written in a single script. Separating different scripts in multi-script documents is an important consideration in document engineering. However, identification of the script will not be sufficient. It is not sufficient for a spell checker to know that the document is in Devanagari script, it needs to know if the language is Sanskrit, Marathi or Hindi. Identifying the language from small text samples is also an important problem in document engineering [2].

3 Characters

3.1 What is a Character?

There are different systems of orthography and their formalization in the form of scripts. English and other European languages are alphabetic in nature - a small set of alphabets are used to compose words using a system of spelling. One must learn to read and write the alphabets as also a plethora of spelling rules. The letters of the alphabets are the characters used in writing the language. Chinese script, on the other hand, is an ideographic script - the script consists of picture elements that signify some meaning. One must learn to read and write the various graphical shapes and their combinations and learn to relate them to the meanings of words. The various graphical elements used in the writing system form the characters of the language.

Indian scripts follow an entirely different scheme - the writing depicts, more or less in a one to one fashion, the various sounds in the languages. The appropriate unit of sound representation chosen is the *syllable.* Thus the word 'Telugu' consists of three syllables 'te', 'lu' and 'gu' and it is exactly these three units that we write when we write the word 'Telugu' in the Telugu script. Indian scripts are syllabic in nature. Thus *there is more or less direct phonetic correspondence between the spoken word and its written representation* - what we write is what we speak. There is no need for rules of spelling at all. This is the most significant merit of the syllabic writing system.

There are, however, some differences between the spoken syllable and its written counterpart. Spoken syllables can be of various types - V, CV, CVC, CVCC, etc. where V represents an vowel sound and C, a consonant sound. However, the written units are always of the C*V kind - zero, one or more consonant sounds followed by an vowel sound. These units of writing are called *akshara*'s. It must be noted, therefore, that the correct terminology for the units of writing in Indian scripts is *akshara*, not syllable, although one finds the term syllable also in literature. In such a case, the distinction between the spoken and the written syllable must be clearly understood. We will always use the term akshara for the units of writing in this paper. *A Character in an Indian script is thus really an akshara.*

3.2 How Many Characters?

Since there are no spelling rules and all spoken sounds must be directly represented in the orthography, the number of possible akshara's is naturally very large. In fact if we allow an unlimited number of consonant sounds in the C*V combination, the number of possible akshara's would become infinite. There is a practical limit to the number of consonants in consonant clusters and the largest consonant cluster known is the 5 consonant cluster in the Sanskrit word 'kaartsnya' involving the consonants 'r', 't', 's', 'n' and 'y' in a single akshara. It can be shown that if all possible C*V combinations upto 5 consonants were allowed, then the number of possible akshara's would still be extremely large - of the order of ten billion! Clearly, not all combinations are possible. Only about 20,000 aksharas are found in the text corpora available in Indian languages. Further, our studies on corpora have shown that about 5000 aksharas account for more than 99% of all the words used in all the major Indian languages.

It is clearly not possible to represent or code so many aksharas directly. Also, a representation scheme needs to be *complete* - every possible akshara must be capable of being represented, irrespective of whether it is frequently used or not, irrespective of whether it occurs in a particular corpus or not. Fortunately, Indian scripts have evolved a unique and ingenious method - a *script grammar*, to achieve elegance, simplicity and economy without compromising on completeness.

3.3 A Grammar for Scripts

Akshara's, (or syllables) are composed of consonants and vowels. A small Finite State Machine is sufficient to recognize and generate all valid aksharas and at the same time prohibit ungrammatical combinations. Note that the notion of grammatically valid and invalid combinations does not exist in other systems of writing. A word may be spelled correctly or incorrectly in English. A word may be a valid English word or it may not be. But there is nothing that makes a spelling ungrammatical. Ungrammatical combinations can never occur in the language, not even in proper names or newly coined words. A script level grammar is unique to Indian languages.

All valid aksharas can be constructed from a small set of less than 100 symbols using the script grammar. There are about 40 consonants and about 15 vowels. All aksharas are composed from these basic units. *Aksharas are the atomic units of writing and consonants and vowels are thus sub-atomic units used to construct a grammar for all valid combinations.* The finite state grammar shown below can generate exactly the (infinite) set of valid aksharas:

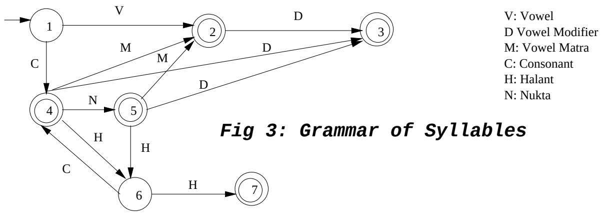

Figure 1: Script Grammar

The following points may be noted with regard to this script grammar. In Indian scripts, the written shapes of vowel sounds when they occur in conjunction with consonants are vastly different from the shapes used to represent pure vowels. In keeping with this significant property, this script grammar employs two sets of symbols, one for pure vowels and the second, called *vowel maatra*s, for the vowel sounds in conjunction with preceding consonants. (Consonants may also show variations in shape depending on whether they occur independently or in conjunction with other consonants. The choice is entirely conditioned by the occurrence of a cluster and these variants are not depicted explicitly in the above grammar.) By definition, vowels can be pronounced independently whereas consonants can only be pronounced along with a vowel sound. It is a well established convention in Indian languages, therefore, that consonants are assumed to have an implicit 'a' vowel in them unless this vowel is removed by an explicit symbol called 'halant'. A consonant without an implicit vowel is called a *pure consonant*. Thus consonant clusters are formed

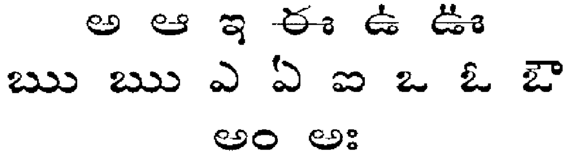

Figure 2: Telugu Vowels

by combing pure consonants. When a consonant combines with a vowel maatra, the implicit 'a' vowel is replaced by the corresponding vowel. (Another possible approach could be to view all consonants as pure consonants by default and add the 'a' vowel just as any other vowel as and when required.) Further, vowel modifiers, which are neither pure vowels nor pure consonants, are accommodated as final components of aksharas. Lastly, the grammar allows not only C*V syllables but also vowel-less segments, which are obtained by a sequence of two halants.

Learning to read and write Indian scripts involves recognizing the individual shapes for isolated and conjunct consonants, vowels, vowel maatras and vowel modifiers. Once these basic shapes are mastered, the script grammar dictates the way aksharas are formed. There is no need to memorize any spelling rules. The figures below depict the basic shapes used in the Telugu script:

4 Fonts and Glyphs

The graphical shapes of the aksharas in Indian scripts are quite complex. Aksharas are of variable width and the vowel maatras and conjuncts embellish the base part on the top, on the right side, left side or at the bottom. Text in Indian scripts is not simply a linear sequence

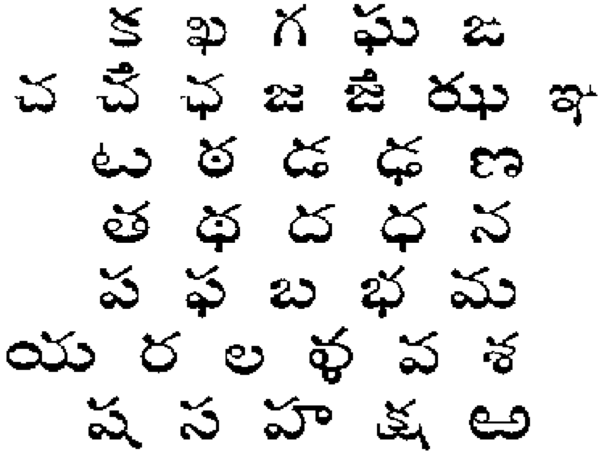

Figure 3: Telugu Consonants

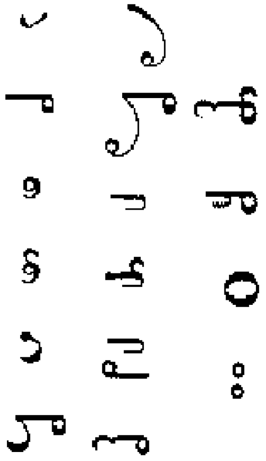

Figure 4: Telugu Vowel Maatras

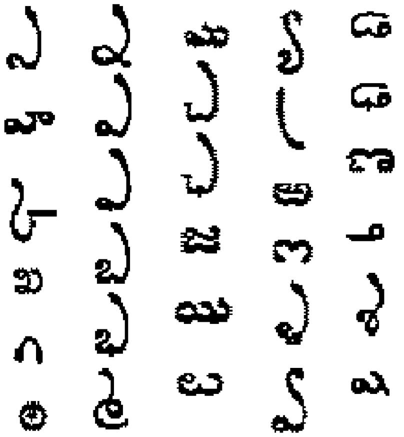

Figure 5: Telugu Secondary Consonants

of more or less equal with and equal height alphabets. This throws up many challenges in the computer processing of Indian scripts and good design calls for thorough understanding and careful consideration of many issues.

We have seen that the number of characters in Indian scripts is very large and therefore cannot be listed or coded directly. The graphical shapes of the aksharas in Indian scripts are quite complex and aksharas can only be composed from smaller, simpler shapes. These elementary shapes used for rendering are called *glyphs.* Glyphs need to be designed for ease of composition into aksharas. The units of the script grammar are not the best choice for the purpose. Indian language fonts therefore use sets of glyphs that do not have one to one correspondence with the script grammar elements. Several symbols may be combined to form a single glyph and a single symbol may actually be composed of several glyphs:

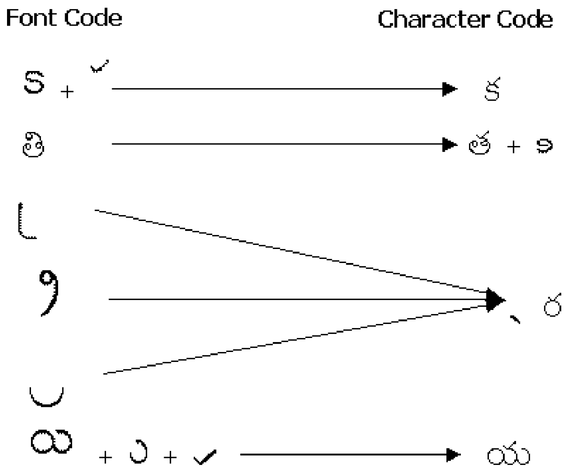

Figure 6: Examples of Font-to-Character Mappings

Further, there may be several glyphs for the same symbol and the choice of the appropriate glyph would be based on the context. For example, the Hemalatha font for Telugu using several glyphs for the 'aa' maatra, one for wide consonants, one for narrow and tall consonants and so on. Choice of the right glyph and appropriate graphical placement and composition are necessary aspects in the design of fonts and rendering. The beauty of fonts lies in the beauty of the glyphs. There are no glyph level standards and the font designers are free to choose the glyphs the way they wish. In fact even the positioning of the glyphs in the font table is also not standardized - Thus each font is different.

A Font is a set of glyphs placed in a suitable format. Glyphs may be defined as bitmaps or, more often, as second or third degree splines. Fonts for Indian languages ideally must also specify the rules for choice of glyphs and for glyph placement and composition. Further, the rules for mapping from character encoding standards must be specified. With TTF fonts, widely used in Indian languages, the mapping from character encoding schemes is completely left unspecified, leading to ad-hoc solutions by commercial vendors of word processors and other software. Unicode fonts attempt to overcome some of these deficiencies but there is still a long way to go [3].

5 Character Encoding Standards: ISCII and UNICODE

We have seen that Indian scripts are phonetic - what we write is what we speak. Orthography is actually a graphical rendering of sounds. In English you write 'es, see, ech, o, o, el' and read it as 'school'. In Indian scripts, you can directly write atomic units for the sounds 'skuu', and 'l'. The basic sounds (phonemes) used in various Indian languages are almost the same - there are very few variations from language to language - sounds are universal. A 'ka' sound is a 'ka' sound irrespective of the wide variations in shape you will see in different scripts. Therefore, if we encode the sounds rather than the graphic shapes used in different scripts, we will get a universal character encoding scheme that captures the essential nature of these languages. This is the essence of the ISCII (Indian Script Code for Information Interchange), a standard of the Bureau of Indian Standards. UNICODE, as we shall see below, is only a minor variation to the ISCII design.

The ISCII standard specifies the script grammar and provides an 8 bit code for the vowels, consonants, vowel modifiers and vowel maatras. There are about 40 consonants and about 15 vowels and the total number of symbols to be encoded is less than 128. So the second half of the ASCII chart is used, keeping the first half in tact. This permits a straight forward mixing of English (Roman) and Indian scripts, a regular phenomenon in the Indian context. The ISCII codes are phonetic based and are therefore the same for all languages. Thus transliteration (change of script) is trivial. All that you need to do is to note the language and use an appropriate font for rendering - the text itself does not change at all. Thus, pure *ISCII documents are script independent representations and the same document can be rendered in any chosen script.*

UNICODE, in keeping with its philosophy of encoding various languages separately, adapts the ISCII system by adding an extra byte for each code to specify the language. This permits multi-lingual plain texts, something that is not possible in ISCII. Plain texts cannot, by definition, include attributes. Thus just one attribute, the language attribute, is implicitly encoded in UNICODE. This may not be a major advantage as any way we will have to go for attributed texts to handle any other attribute. However, ISCII is not widely supported directly by Operating Systems, Browsers etc. and UNICODE will surely score in this regard. There are still teething problems in the adaptation of UNICODE for Indian languages and scripts [3]

6 Text Representation and Processing

ISCII is not directly supported by Operating Systems or Web Browsers. That is, ISCII encoded documents cannot be directly viewed - they need to be rendered in a suitable font. There are no glyph level standards and thus mapping from ISCII to a given font becomes a critical step. Aksharas are composed of more basic units at both the character encoding level and the font encoding level but the two schemes are completely different. Most commercial software vendors have chosen to directly encode documents in their proprietary and non-standard fonts. Thus there is no compatibility between different software products. Font encoded documents cannot be considered as texts and no standard text processing can be performed on such documents. This single factor has contributed to a colossal waste of time and effort in document engineering in India.

A variety of solutions have been proposed for mapping character encoded documents into a specified font. Plug-ins have been developed but this is by its very nature an ad-hoc solution - a plug-in is software and version specific. Context Free Grammars have been used to obtain a translation system on line with traditional compilers. Most commercial software systems use hard coded rules. Some have resorted to near-exhaustive enumeration. None of the systems in existence today are complete (that is they properly handle all valid aksharas), consistent, or computationally efficient. A consistent and provably complete system based on an Augmented Finite State Transducer is being developed at the University of Hyderabad.

Switching to UNICODE will solve many of these problems but there will still be some issues wanting to be addressed properly. The distinction between language, script and font is not as clear as it should be in the UNICODE world. For example, the script grammar is an innate property of a script and it cannot be ignored or indirectly addressed at the level of fonts. A UNICODE encoded text is still simply a sequence of codes and the notion of an akshara s an atomic unit is not explicitly evident. UNICODE based system are yet to pick up in India for a variety of reasons. This is not a small change for many who have still been using font encoded documents. UNICODE fonts are not freely available and OS and Browser support is only slowly becoming available.

Given this scenario, avoidable multiplicity of spellings, splitting of aksharas across line boundaries, incompatibility between systems and between fonts, poor or no support for even basic operations such as searching and sorting etc. continue in word processors, web documents and other such systems. Fundamentally sound designs such as the AKSHARA system developed at University of Hyderabad are expected to mitigate these problems in the near future.

7 DRISHTI: An OCR Engine

Here we describe briefly the architecture of an OCR system for Telugu script called DRISHTI, which highlights the concepts described earlier. DRISHTI as a system was first described [5], and subsequently improved [1]. Presently DRISHTI handles good quality Telugu documents with a single column of text scanned at 300 dpi.

As reported [1] it has a recognition accuracy at the glyph level of about 97%. It has been tested on several kinds of input ranging from newspapers, popular novels to laser printed text.

As explained earlier, there can be a large number of aksharas (C*V combinations), which need to be recognized. In DRISHTI this complexity of training and recognition is brought down to just a few hundred units, by identifying connected components. Essentially the compositional approach is used where connected components from the binarized image are the units which are recognized. This bypasses tricky issues of attempting to segment maatras from the base characters. Fortunately secondary consonants in clusters are distinct connected components in Telugu and they need not be dissected from their main consonants like in Devanagari or Bangla scripts. This idea has also been used by other approaches to Telugu OCR [8], [6].

Line, word and character separation are the stages in isolation of the connected components which are then fed to the recognition engine. Telugu characters are mostly rounded and devoid of straight strokes. Simple projection profiles are not suitable for the complexity of Telugu orthography. A smearing technique using RLSA (Run-Length Smearing Algorithm) [9] with both vertical and horizontal thresholds estimated from the document are used. This helps to form words and lines. Connected components are then isolated from the words found by smearing. The layout analysis yields position information.

Recognition of connected components is done using advanced template matching with fringe distances. Templates of a standard size are matched with input scaled to the template size of 32x32 pixels. Fringe distance matches were suitably improved by use of distance techniques for binary template matching as described by Tubbs [7].

The complete OCR algorithm is given below:

1. Read in an input binary image
2. Segment the image into lines and words
3. Extract the connected components from each word
4. For each component
   (a) Normalize size to match stored templates
   (b) Compute fringe distance map
   (c) Compute fringe distance from all templates
   (d) Output template with smallest fringe distance
   (e) Convert template code to ISCII code
5. Store ISCII output in a file

Here we should note that the step 4(e) above is not at all a trivial task. This step is perhaps unique to the Indian OCR systems. We describe the details of this process in the following section.

8 From Recognition to Text

The OCR engine outputs codes corresponding to connected components recognized. There as as many codes as there are different connected components. These codes need to be mapped onto a character encoding standard such as ISCII or UNICODE before it can be viewed and edited as text. There are several factors that make this text reconstruction process quite complex:

- The DRISHTI OCR engine recognizes connected components, which do not necessarily correspond to aksharas or the building blocks of aksharas - vowels, consonants, vowel maatras etc. A connected component may correspond to one building block or fractions or multiples thereof.

- A code output by the core engine may actually stand for several different things. The shapes of certain consonants are same when used as base characters or as secondary consonants in clusters. The codes output by the OCR engine will be the same in both cases but their character level representations will be different - a preceding halant needs to be inserted when the consonant is a secondary consonant in a cluster.

- In some cases, the presence of a connected component implies a change in the character recognized without it.

- The order which the OCR engine outputs the codes for the connected components does not necessarily correspond to the order in which aksharas are composed. The OCR engine basically works left to right and top to bottom on the scanned image. Parts of one akshara may get mixed up adjacent aksharas.

- Line and word segmentation may not be in error, there may be mis-recognitions as also rejects.

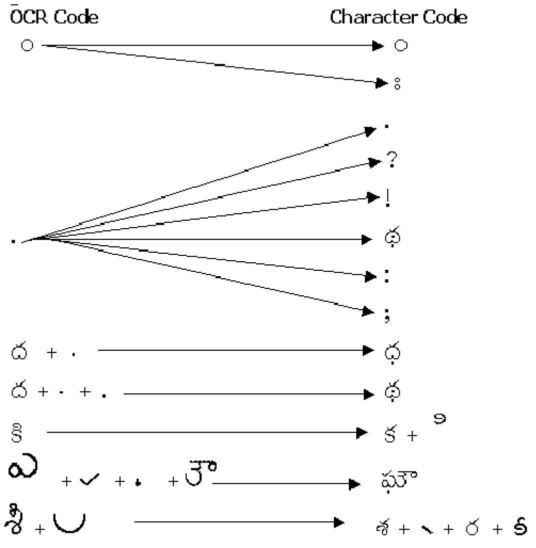

Figure 7: Examples of OCR Output Code to Character Mappings

The following figure illustrates some of these problems:

We now present the overall strategy used in the DRISHTI OCR system for Telugu. The core engine is designed to output not only the codes for the recognized components but also the sizes and relative locations of the components. The size and relative location and vertical stacking information aid in the disambiguation and proper re-combination of base and conjunct consonants, punctuation marks, etc. Accordingly, recognized components are classified into base character, secondary consonant in a consonant cluster, vowel maatra and punctuation. Identification of base characters has been found to be very robust. There are only a few known exceptions, all of which can be disambiguated through the corresponding the character level codes. As a rule, aksharas start with a base character (full vowel or a consonant) and this rule is used to hypothesize syllable boundaries. Once syllable boundaries are determined, the script grammar is used to re-order the other components to build valid aksharas. This overall strategy has been found be working quite well.

9 Conclusions

We have described a script grammar which defines a framework for the representation and processing of Indian language documents at the language, script and font levels. The case study of an OCR system within this framework for Telugu instantiates the general proposition. This abstraction helps to get a deeper understanding of complex systems for document engineering.

References

[1] C. Bhagvati, T.Ravi, S.M.Kumar, and Atul Negi. On developing high accuracy ocr systems for telugu and other indian scripts. *Proceedings Language Engineering Conference, Eds. (K.N. Murthy, BB Chaudhuri)*, IEEE Computer Society Press(2003):pp 18–23, 2002.

[2] G Bharadwaja Kumar and Kavi Narayana Murthy. Script independent language identification in the indian context. In R M K Sinha and V N Shukla, editors, *Proceedings of iSTRANS 2004 International Conference - Vol 1*, pages 74–81. Tata McGraw-Hill Publishing Company Ltd, 2004.

[3] Narayana Murthy Kavi. Issues in standardization of character encoding schemes. *Technical Report*, LERC/UoH/2002/1(Department of CIS, University of Hyderabad), 2002.

[4] Narayana Murthy Kavi and Nandakumar Hegde. Some issues relating to a common script for indian languages. *International Conference on Indian Writing Systems and Nagari Script*, 6-7 February 1999(Delhi University, Delhi, India), 1999.

[5] Atul Negi, Chakravarthy Bhagvati, and B. Krishna. An ocr system for telugu. *Proc. Sixth International Conf. Document Analysis and Recognition. Seattle, USA*, IEEE Computer Society Press, Los Alamitos, CA(2001), 2001.

[6] Atul Negi, K. Nikhil Shankar, and Chandrakanth Chereddi. Localization, extraction and recognition of text in telugu document images. *Proc.Seventh International Conf. Document Analysis and Recognition. Edinburgh, Scotland*, IEEE Computer Society Press, Los Alamitos, CA(2003):pp 1193 – 1197, 2003.

[7] J.D. Tubbs. A note on binary template matching. *Pattern Recogition*, 22(4):pp 359–365, 1989.

[8] C. Vasanthalakshmi and C. Patvardhan. A multi-font ocr system for printed telugu text. *Proceedings Language Engineering Conference, Eds. (K.N. Murthy, BB Chaudhuri)*, IEEE Computer Society Press(2003):pp 7–17, 2002.

[9] K. Y. Wong, R. G. Casey, and F. M. Wahl. Document analysis system. *IBM Journal of Research and Development*, 26(6):pp 647–656, 1982.