Classification and Identification of Telugu Aksharas using Moment Invariants and C4.5 Algorithm

C. Srikanth, B.L. Deekshatulu, C.Raghavendra Rao
and Chakravarthy Bhagvati

A.I. Laboratory, Department of Computer and Information Science,
University of Hyderabad – Hyderabad -500046, India.

Abstract

Classifying and recognizing Telugu characters (aksharas) is a challenging task because of the variations in the script and the large number of characters. The complexity of the shape is a result of structural compositions involving vowels (V), consonants (C), consonants with vowel modifiers (CV) and consonant clusters (CCV). This paper presents a novel classification strategy for classifying aksharas with the CV structure. This is achieved by constructing two decision trees — one for consonants and the other for vowel modifiers — by hybridizing the moment invariants and C4.5 algorithm. The results, although preliminary and limited in scale, illustrate the potential strengths of the approach. Perhaps, the uniqueness and the strength of the approach lies in the divide-and-conquer strategy where the two orthographically orthogonal decision trees each recognizing only a few tens of classes may in combination potentially classify hundreds of classes.

Introduction

Optical character recognition (OCR) refers to the field of study dealing with converting the character images present in a scanned document image into computer readable text. Usually, the output from an OCR system is text represented in ASCII for English and Unicode or ISCII for Indian scripts.

Research on OCR systems has a long history for English [1] and several commercially successful systems, such as Abby FineReader, are in existence today. The situation is different for Indian scripts. Although initial and pioneering papers appeared in the '70s, it is only in the past decade that truly systematic development has been taking place with assistance from the Govt. of India for developing technology solutions for Indian languages [2]. The results are indicated in a number of papers on Indian OCR systems in recent ICDAR conferences [3]–[7].
OCR techniques reported in literature view the problem as one of classification, i.e., assigning a label representing a character to a pattern of pixels comprising an image of the character. One of the key components of such an approach is classifier design. The challenge lies in the large character sets seen in many Indian scripts and therefore the need for a large multiclass classifier. Whereas the English script has less than a 100 different symbols (lower and uppercase characters, punctuation and special symbols), most Indian scripts involve classifying a few hundred and for some languages (Hindi and Bangla) more than a thousand characters. All the traditional classifiers such as nearest neighbour [5], hand-crafted decision trees [8], neural networks [9] and SVMs [10] have been tried with varying levels of success.

In this paper, we present a different idea of classifying and recognizing Telugu characters using a combination of two decision trees. One decision tree is trained for recognizing consonants (C) while the other is trained for vowel-modified consonants (CV). The approach is based on hybridizing the set of seven invariant moments proposed by M. K. Hu [14] and the C4.5 algorithm of J. Ross Quinlan [11].

The rest of the paper is organized as follows. We present the main characteristics of the Telugu script in Section 2 followed by our approach in Section 3. Section 4 summarizes the results while in Section 5, we present our conclusions.

Telugu Script
Telugu is the main language of Andhra Pradesh State in India and records indicate its existence in almost the present form since 7th Century AD. Telugu is a phonetic language with an ortho-syllabic script that is written from left to right, with each character generally representing a syllable. Telugu alphabet is generally said to have 16 vowels, 36 consonants and three special symbols. Modern Telugu consists, however, of only 12 vowels, 36 consonants and 2 special symbols (shown in Figure 1). Besides these, Telugu script consists of  (a) Vowels(12)  

\[
\begin{align*}
&\ddot{a}, \ddot{e}, \ddot{i}, \ddot{u}, \ddot{\text{a}}, \ddot{\text{e}}, \ddot{\text{i}}, \ddot{\text{u}}, \ddot{\text{A}}, \ddot{\text{E}}, \ddot{\text{I}}, \ddot{\text{U}}, \\
&\dddot{a}, \dddot{e}, \dddot{\text{a}}, \dddot{\text{e}}, \\
&\ddot{\ddot{a}}, \ddot{\ddot{e}}, \ddot{\ddot{\text{a}}}, \ddot{\ddot{\text{e}}}, \\
&\dddot{\dddot{a}}, \dddot{\dddot{e}}, \dddot{\dddot{\text{a}}}, \dddot{\dddot{\text{e}}},
\end{align*}
\]

(b) Consonants (36)

\[
\begin{align*}
&\ddot{\ddot{\text{r}}}, \ddot{\ddot{\text{R}}}, \\
&\dddot{\dddot{\text{r}}}, \dddot{\dddot{\text{R}}}, \\
&\ddot{\ddot{\text{g}}}, \ddot{\ddot{\text{G}}}, \\
&\dddot{\dddot{\text{g}}}, \dddot{\dddot{\text{G}}}, \\
&\dddot{\dddot{\dddot{\text{r}}}}, \dddot{\dddot{\dddot{\text{G}}}}, \dddot{\dddot{\dddot{\text{g}}}}, \dddot{\dddot{\dddot{\text{G}}}}
\end{align*}
\]

(c) Special symbols(2)

\[
\begin{align*}
&\dddot{\dddot{\text{J}}}, \dddot{\dddot{\text{D}}},
\end{align*}
\]

Figure 1: The Telugu alphabet.
other symbols called *vattulu* which are half-consonants that appear in consonant clusters (CCV). Vowel marks (VM) or half-vowels are also used in modifying a consonant with a vowel sound. Examples are shown in Figure 2 where the first character represents the sound *ku* and the vowel mark

\[ \hat{\text{k}}, \hat{\text{u}} \]

**Figure 2:** Examples of compound characters: the first has a CV structure while the second has a CCV structure.

The second character in the Figure is the sound *kri* where the circular stroke at the bottom is the *vatti* corresponding to the consonant \( \hat{\text{k}} \) and the little circular mark at the top is the vowel mark corresponding to the vowel \( \hat{\text{u}} \). The complete sequence of vowel modified variants of the consonant \( \hat{\text{k}} \) is shown in Figure 3. Such a sequence is called a *gunintham*.

\[ \hat{\text{k}}, \hat{\text{i}}, \hat{\text{u}}, \hat{\text{g}}, \hat{\text{u}}, \hat{\text{o}}, \hat{\text{a}}, \hat{\text{y}}, \hat{\text{u}}, \hat{\text{a}}, \hat{\text{v}}, \hat{\text{u}}, \hat{\text{r}}, \hat{\text{o}}, : \]

**Figure 3:** The *ka gunintham* showing the full set of vowel marks.

A *gunintham* defines a natural class in Telugu language and we train our classifier to recognize the different characters within a gunintham as a single class. All the vowels are grouped into a single class (labelled ‘1’). The consonants starting with \( \hat{\text{k}} \) and ending with \( \hat{\text{p}} \) are assigned the labels ‘2’ to ‘37’. The vowel marks are assigned the labels ‘1’ to ‘16’ for the full set of 16 modifiers that appear in older texts.

**Our Approach**

Our approach is to classify a given unknown character into its base consonant class using one decision tree and then into a specific vowel mark class using the second decision tree. The combination of the two gives the precise label for the character. The features used for both the decision trees are *invariant moments*. The first decision tree needs to classify a character into 36 classes while the second outputs one of 16 classes. By using group classification based on guninths and combining the output from two relatively small decision trees, which in some sense are orthogonal in their functionality, we achieve a combined classification capability of 576 classes (36×16). It is this aspect of our classifier design that is most interesting.
A. Moment invariants

Moment invariants are well-known in literature and characterize properties of connected regions in binary images. These moments are invariant to translation, rotation and scaling and hence their name of invariant moments. They are useful because they provide a fairly simple representation of shape for classification and recognition tasks. The seven invariant moments are derived from the normalized central moments

\[ \eta_{pq} = \frac{\mu_{pq}}{\mu_{00}} \]

where

\[ \gamma = \frac{p + q}{2} + 1 \]

for \( p+q = 2, 3, \ldots \), and \( \mu_{pq} \) are the standard central moments of 2-D functions. The set of seven invariant moments are defined as

\[
\begin{align*}
\phi_1 &= \eta_{20} + \eta_{02} \\
\phi_2 &= (\eta_{20} - \eta_{02})^2 + 4\eta_{01}^2 \\
\phi_3 &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \\
\phi_4 &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \\
\phi_5 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) - 3(\eta_{21} + \eta_{03})^2 + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03}) \( \text{[Alternative form]} \) \\
\phi_6 &= (\eta_{30} - \eta_{02})(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2 + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \\
\phi_7 &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2 + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03}) \( \text{[Alternative form]} \)
\end{align*}
\]

The above definitions are taken from Gonzalez [12].

Table I gives the seven invariant moments for three sample characters $\ddot{\varsigma}$ ($na$), $\dddot{\varsigma}$ ($ki$) and $\dddot{\ddot{c}}$ ($lo$). The column titled C-Code gives the consonant code while the column titled CV-code gives the vowel mark code. From the table, it may be seen that $\ddot{\varsigma}$ is represented by the two code sequence ‘21’ and ‘1’ while, $\dddot{\varsigma}$ is represented by ‘2’ and ‘3’ and $\dddot{\ddot{c}}$ is ‘29’ and ‘13.’ Also, it should be clear now that the sound ko (the character $\dddot{\ddot{c}}$) would be represented by ‘2’ and ‘13.’

Table I: Seven invariant moment values for sample characters.
B. Data and sample decision tables

C4.5 is a supervised learning algorithm used to construct decision trees from the given data. It is an extension of ID3 algorithm [11] and is not restricted to binary splits and uses simple depth-first construction.

Training data had been generated from Andhrabhoomi newspaper after suitable preprocessing to remove skew, noise and perform binarization. Invariant Moments (IM) for each character had been calculated. The IM values thus generated are discretized in order to fit into the C4.5 algorithm. The discretization had been done using \[ \frac{x - \mu}{h\sigma} \] where \( x \) is the IM value, \( \mu \) is the mean and \( \sigma \) is the standard deviation and \( h \) is a scaling constant.

The mean and standard deviation are calculated from the training set and used for discretization. The discretized values of the seven moments for the same characters shown in Table I are given in Table II.

These discrete values are input to the C4.5 algorithm to construct a decision tree for classifying an unknown character into a base consonant (i.e., the C-code). Sample rules for

\[ \text{Table II: Discretized moments for the sample characters shown in Table I.} \]

\[
\begin{array}{cccccccc}
\text{CH} & \text{DM1} & \text{DM2} & \text{DM3} & \text{DM4} & \text{DM5} & \text{DM6} & \text{DM7} \\
1 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\
2 & 3 & 3 & 3 & 4 & 4 & 4 & 3 \\
3 & 4 & 4 & 4 & 5 & 4 & 4 & 4 \\
\end{array}
\]

Some instances of the guninthams or vowel-modified versions of the three consonants shown in the previous tables are shown below.

\[ \text{Table III: Sample rules for various instances of the three consonants shown in previous tables.} \]

\[
\begin{array}{cccccccc}
\text{CH} & \text{DM1} & \text{DM2} & \text{DM3} & \text{DM4} & \text{DM5} & \text{DM6} & \text{DM7} \\
2 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\
2 & 3 & 3 & 3 & 4 & 4 & 4 & 3 \\
2 & 4 & 4 & 4 & 5 & 4 & 4 & 4 \\
2 & 4 & 4 & 4 & 5 & 4 & 4 & 4 \\
2 & 4 & 4 & 4 & 5 & 4 & 4 & 4 \\
2 & 4 & 4 & 4 & 5 & 4 & 4 & 4 \\
\end{array}
\]

Table III is based on the output generated by Matlab for consonant recognition. The column order shows the relative importance of each discretized moment roughly according to entropy gain while the first column shows the character class code (C-code). The first moment has the highest entropy gain. Then, depending on the next
dominating entropy value, it generates a complete tree with the decision attribute as the leaf node. The table should be interpreted as rules in the following manner. For example, the rule in the first row of Table III is that

if the value of discretized moment 1 is ‘3’ and the value of the discretized moment 7 is ‘3’ and the value of discretized moment 6 is ‘4’ and ... and the value of discretized moment 4 is ‘3’, then the sample character belongs to the class ‘2’, i.e., $\xi$ (ka).

A similar approach is followed for constructing a decision tree to recognize the vowel-marks or guninthsams. From the resulting tree, it was found that the third discretized moment (DM3) has the greatest entropy gain in this case with the first discretized moment coming up in the third place.

Figure 4 shows a portion of the decision tree for classifying consonants. It is generated from the Matlab output shown in Table III.

**Validation Tests and Results**

The Telugu characters for both training and testing are taken from 10 editorials of the *Andhra Bhoomi* newspaper.

![Figure 4: A portion of the decision tree constructed for classifying consonants.](image)

Several sets of experiments were done using a 10-fold test [11]. The different experiments were done by varying the value of $h$, the scaling constant in the discretization function. *CART* algorithm [11] is also tried for comparison. It has generally been found that C4.5 gives better results than CART. The results of these initial experiments are summarized in Table IV. The number of distinct characters
observed in the test sets is approximately 350 (which may be compared against the total possible characters which is $36 \times 16$ or 576).

It may be seen from Table IV that the classification accuracy does not vary much for $0.5 \leq h \leq 1.0$. Also, CART algorithm does not depend on $h$ and therefore the results are simply replicated for the different values of $h$.

Later, a further set of editorials were used in training and testing. The results of a 10-fold test showed that the accuracy is 98.52% for characters with CV structure and 97.03% for the C structure. These results may seem contradictory to intuition because a base consonant should be simpler in shape when compared to a vowel-modified consonant. A possible explanation is that several consonant-vowel-modifier combinations do not occur with sufficiently high frequency. A case in point occurs with consonants such as $\dddot{g}$ and $\dddot{y}$ that may be easily confused or mis-recognized. However, the vowel-modified forms of the consonant $\dddot{y}$ are extremely rare when compared with that of $\dddot{g}$ which leads to less number of confusions when the CV structure is used.

It may also be noted that a distinguished Telugu linguist Dr. Bh. Krishnamurti notes that the CV structure is the most common character in Telugu comprising roughly 78% of all the characters in large corpora [13]. Consonants make up a significant part of the remaining characters while vowels, which can occur only at the beginning of a word, and CCV structures are relatively less frequent. Therefore, designing a classifier or an OCR system for Telugu may well give a higher priority to C and CV structures for achieving high accuracy.

### Table IV: Results on initial training and testing on a dataset obtained from 10 *andhra bhoomi* editorials.

<table>
<thead>
<tr>
<th>CH TYPE</th>
<th>$h$</th>
<th>C4.3%</th>
<th>CART%</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>1.0</td>
<td>95.1</td>
<td>89.57</td>
</tr>
<tr>
<td>CV</td>
<td></td>
<td>97.55</td>
<td>89.55</td>
</tr>
<tr>
<td>C</td>
<td>0.5</td>
<td>93.82</td>
<td>89.66</td>
</tr>
<tr>
<td>CV</td>
<td></td>
<td>96.13</td>
<td>89.55</td>
</tr>
<tr>
<td>C</td>
<td>0.25</td>
<td>91.04</td>
<td>89.66</td>
</tr>
<tr>
<td>CV</td>
<td></td>
<td>93.62</td>
<td>89.55</td>
</tr>
</tbody>
</table>

Conclusions
In this paper, we showed that shape-based features such as moment invariants in combination with a well-designed classification strategy lead to high accuracy in recognizing Telugu characters. The results reported are no doubt on small datasets but the approach is interesting. The strategy of dividing the character recognition problem comprising several hundred classes into two ‘orthographically orthogonal’ decision trees is perhaps unique. That neither of the two decision trees recognizes more than a few tens of classes gives it additional beauty in that the training sets and feature sizes may be
relatively small. In our approach and experiments, the two decision trees, using seven invariant moments, and classifying 36 and 16 classes respectively have together recognized nearly 350 distinct characters (hence class labels) with a good accuracy.

References