OCR for Telugu Script

Monis Rakesh Ajith

December 16, 2011

Abstract

The unavailability of an Optical Character Recognition (OCR) scheme meeting basic standards has been a major impediment in the digitization of Telugu. The quest for an appropriate solution has been elusive due to the inherently complicated nature of the script itself and partly to the lack of a concerted effort. We propose a scheme based on K-means and Support Vector Machines that improves accuracy of current implementations. We have performance tested against deliberately introduced sources of errors which are known to cause performance issues while lacking robust solutions.

Introduction

Telugu script is an abugida written in syllables; each unit being a combination of at least one constant and associated vowels being attached laterally attached vowels. Each syllable is a combination of one or more consonants and laterally attached vowels.This contiguous nature of the script makes it impractical to separate characters. Consequently training the recognition algorithm to recognize a larger set of symbols may be more robust rather than a messier approach requiring extraction of relative glyph positions. Different permutations of the 16 vowels and 37 consonants give rise to approximately 500 unique glyphs. For this project we focus on the 471 most commonly used glyphs. Despite its popularity of use (over 70 million native speakers) there are no OCR schemes available which could be deployed commercially.

Rendering

Stanford as rendered in Telugu is depicted below. Notice the rendered combination of a vowel and two preceding consonants, with the vowel attached above and the second consonant below.

Feature Learning

When viewed through the lens of accuracy; feature learning is a critical aspect in any OCR algorithm especially considering the low accuracy which was achieved by an Artificial Neural Networks(ANN) based implementation that we initially attempted. Though this ANN algorithm may not have been accurate enough in itself, it did provide us with important insights into developing an accurate and robust framework.

Auto encoders are the method of choice for the reduction of higher dimensional pixel-data to a lower but much more tractable feature space. Papusha and Satheesh provide some novel ideas of extracting features using K-means, with results similar to auto-encoding techniques while achieving much higher agility in terms of both implementation and execution. Eventually a support vector machine (SVM) based solution is seen to offer higher accuracy.

Training and Testing data.

Glyphs rendered using the popular Pothana font are stored as a single file with identity data and location coordinates made available in a text file. Features are extracted from these relatively pristine digital renditions as part of data pre-processing and are used to score images that need to be identified.

Morphed versions of these digitally rendered images also form the base data set for training and testing. To reflect real-world data, randomly selected pixels are dropped from the image data. Missing pixels, a considerably complex issue that plagues OCR schemes, comparatively affects accuracy to a higher degree than other issues like noise which may easily be resolved using standardized approaches. More importantly it has to be handled as part of the OCR implementation itself while noise reduction, for example, may be handled externally using robust algorithms available as free-ware. Up to 5 degrees of tilt has been randomly introduced to simulate scanning issues that may occur with actual real world data sources. Ultimately each digitally rendered character is modified independently a total of ten times with eight instances being used for training and two for cross validation.

Original rendered image of letter ‘a’ (as in the word Apple) in Pothana font along with three distorted versions.

Base Algorithmic Framework:

PreProcessing:

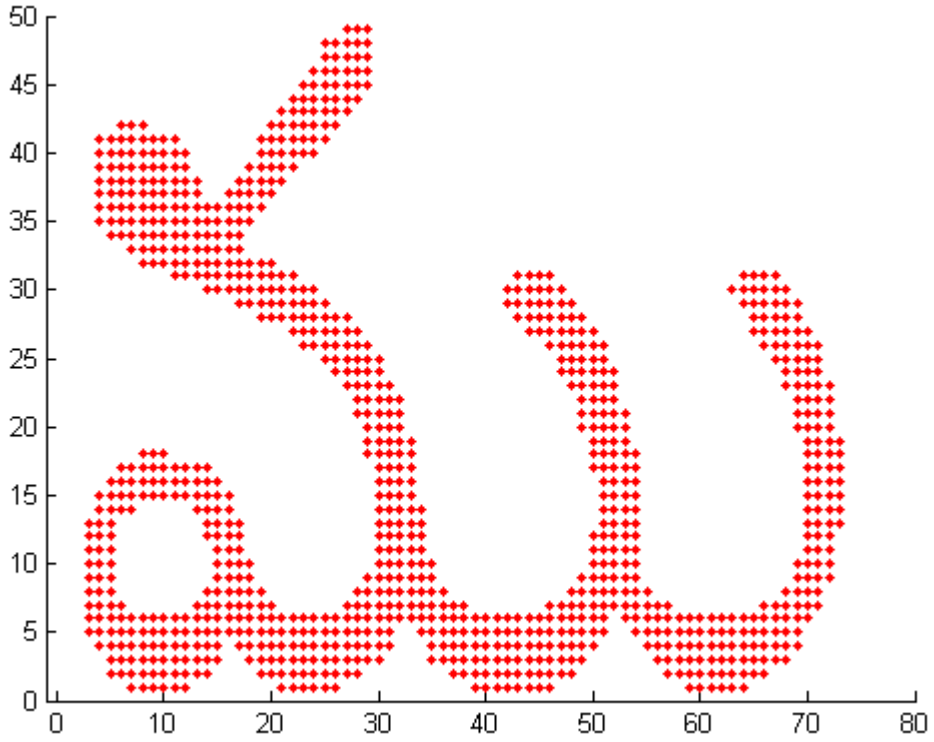
Step 0:
Original input image data is scaled down to a size of 64x64 pixels.

Step 1:
Scaled Images are consequently clustered and over 20,000 pieces of the script from various letters. This is done in either of the two following ways; comparatively depicted in the

Step 1a:
A k-means based clustering approach applied on each glyph.

Step 1b:

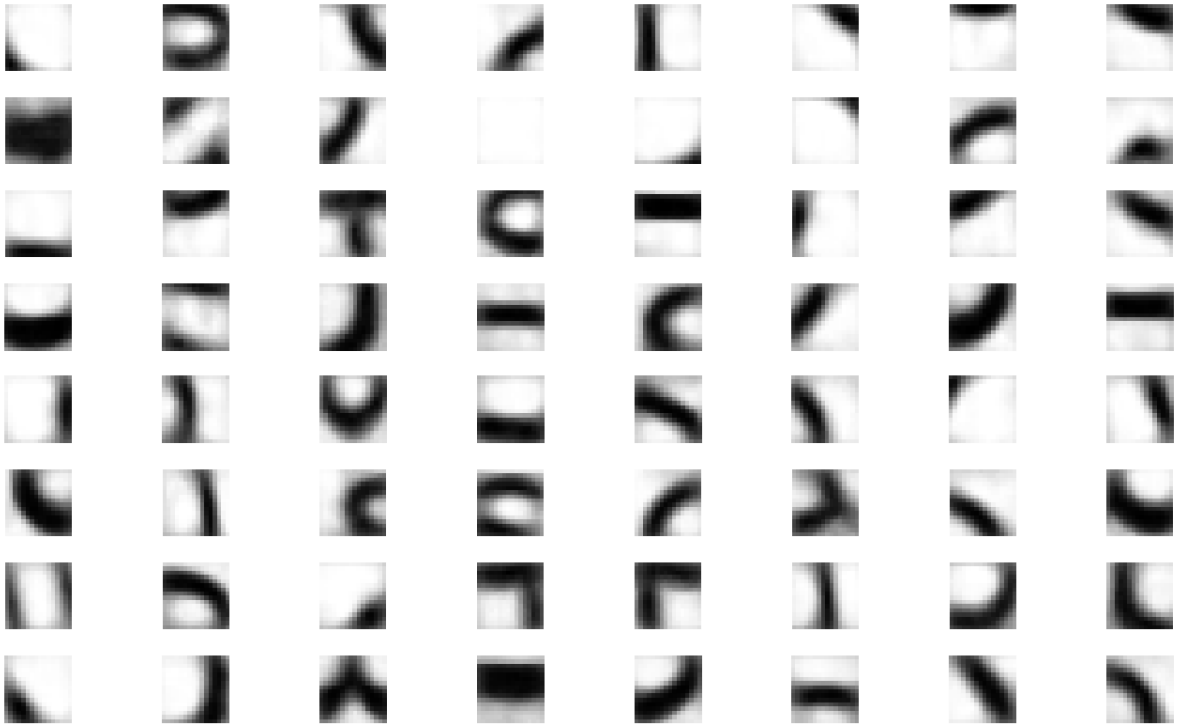
Each of the glyphs are divided in to 50 over lapping pieces which are 16x16 pixels dimensionally.



Step 2:
The 'principal components' of the script are obtained from the above 20,000 pieces by another level of clustering implemented using a k-means based approach (in the 256 dimensional space).

The principal components that define the script, obtained using this approach are equivalent to finding auto-encoding blocks for the script. Both euclidean and hamming distances are implemented, though statistically significant differences in the accuracy between either implementation are not observed.

Sixty four centroids which form the base features are extracted from the training data.These are shown in the image below.



64 centroid or base-features

Feature extraction.

Morphed versions of the digital renditions are used as the input data requiring classification which would be obtained from scanned images in an actual application. These images are divided into four blocks labelled as Top Right, Top Left, Bottom Right, and Bottom left respectively. Each of the four split-image part are processed and distance metrics from the 64 base feature or centroids are calculated. This was done using three different measures (inner product, hamming distance or L2 distance). Any measure may be used as each of the measures provides similar accuracy scores with no statistical significance observed in the scoring-accuracy differences between pairs of the three measures.
These are calculated by processing each scanned image's 4 sub-blocks against each of the 64 centroids. Thus for any given input image after pre-processing (64x64 pixels), for every block (32x32 pixels) and against each centroid (16x16 pixels) we record an average of the distance metric (or inner-products) as a vector denoted as the feature vector.

The feature vector length = number of centroids X number of split-image blocks of the image
length(vector) = 64 X 4 = 256 elements

The aspect ratio of the original scanned glyph is added as the last element that is the 257th element of the feature vector though this must be done before the down-scaling of the glyph's image.

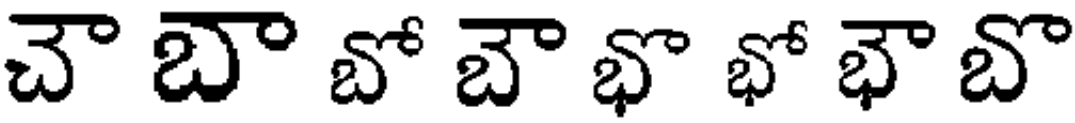
Thus for each of the ten image per glyph we have a 257 element vector. Training the lib-linear SVM is done using 8 of these vectors; the remaining 2 being used for testing.

Results

Out of the nearly 940 test images, 110 were misclassified, resulting in an accurate identification 88% of the time. This is much better than the vanilla ANN that was trained for the mid-quarter milestone.
We hope to improve this accuracy by optimization over arbitrarily chosen parameters like block size and features (or centroids) extracted.

An example of character miss-classification is shown below to illuminate the difficulties that still need to be overcome. Understandably even we had a difficult time as children in being required to distinguish them! Our endurance gradually built up by repeated testing and training.

<<Confused.png>>



Future Work

There are several improvements that we hope to test in the near future in our quest to develop a robust open-source OCR scheme for Telugu other than just optimizing over the number of features extracted and image divisions mentioned above.
Instead of averaging the distances, using the maximum along with its location in the sub-image may lead to an improvement in the accuracy.

Leveraging the number and relative position of consonants and vowels in the image will narrow down the number of centroids in the scoring set for each image. This would require classification of each image on the basis of the number and position of vowels and consonants in the scanned image.

Acknowledgements
This project builds on work done by Sanjeev Satheesh during Fall 2010. We are grateful for his advice and his work provided the inspiration behind some of the aspects implemented specially in the case of the photo OCR. We are also grateful to the people at LIB-LINEAR.

Last but not the least we would like to take this opportunity to acknowledge and thank the guidance that was provided to us by Prof Andrew Ng who not only pointed us in the right direction but also provided a framework to work with in, and the spirit of innovation that is inculcated as part of the class.

References.

Engineering Dayalbagh Educational Institute, Agra, India, “A High Accuracy OCR System for Printed Telugu Text”.

Sheetalashmi R. Abnikant Singh, Department of Electrical Engg, IIT Kanpur, “Optical Character Recognition for Printed Tamil Text using Unicode”.

R. Jagadeesh Kannan, RMK Engg College, Chennai, India, “A Comparative Study of Optical Character Recognition for Tamil Script”.

Sang Sung Park,Won Gyo Jung,Young Geun Shin, Dong- Sik Jang, Department of Industrial System and Information Engineering, Korea University, South Korea, “Optical Character System Using BP Algorithm”.

Ahmad M. Sarhan, and Omar I. Al Helalat, “Arabic Character Recognition using Artificial Neural Networks and Statistical Analysis”.

Princess Summaya University for Science and Technology, Amman, Jordan, “Online Handwritten Character Recognition Using an Optical Backpropagation Neural Networks”.