OCR FOR TAMIL LANGUAGE

A PROJECT REPORT

submitted by

PRADEEP M

(CB.EN.U4CSE08131)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

AMRITA SCHOOL OF ENGINEERING, COIMBATORE

AMRITA VISHWA VIDYAPEETHAM

COIMBATORE 641 112

APRIL 2012

AMRITA VISHWA VIDYAPEETHAM
AMRITA SCHOOL OF ENGINEERING, COIMBATORE

BONAFIDE CERTIFICATE

This is to certify that the project report entitled **”OCR FOR TAMIL LANGUAGE”**, submitted by **”PRADEEP M”,”RegNo.CB.EN.U4CSE08131”** in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in **”COMPUTER SCIENCE AND ENGINEERING”** is a bonafide record of the work carried out under my guidance and supervision at Amrita School of Engineering, Coimbatore.

Mrs. HEMA P MENON, Project Advisor, Assistant Professor, CSE	Dr. K P SOMAN, External Advisor, Professor, CEN

Chairperson, CSE
Prof. P N KUMAR

This project report was evaluated by us on ....

INTERNAL EXAMINER	EXTERNAL EXAMINER

Acknowledgements

I would like to thank my project advisors namely, Mrs. Hema P Menon and Dr. K P Soman for their advice and patience.

I also extend my gratitude to Professor. P N Kumar, Chairperson, CSE. I would like to thank our project co-ordinators namely Mr. M Senthtil Kumar, Mr. A K Sumesh and Mr. R Gowtham.

I also want to thank my collaborator Mr. Praveen Krishnan for providing help and sharing his experience on working in Tamil OCR. I also owe my thanks to parents and friends. In addition, I would like to thank Dr. Sabarimalai Manikandan for giving me an opportunity to work in Tamil OCR.

Finally I want to thank the Almighty.

Abstract

Optical Character Recognition (OCR) system is robust for Latin languages but it is not so for Indic languages. It is partly because of the number of character classes and close similarity of characters.

In this work, we are presenting a comparative study of recognition based on supervised and unsupervised learning. The former is based on SVM based training and classification (SVMTC) and the later is based on Random Projection Technique (RPT). The challenging part of this project is feature extraction. We have extracted features based on active contour model, character geometry, moment invariants, random projection and Gabor filter. It is a combination of statistical and visual features. Overall, RPT gives better results than SVMTC. The result based on RPT is found out to be 58 %.

Table of Contents

Page

Chapter

List of Figures

List of Tables

List of Abbreviations

ACM - Active Contour Model

OCR - Optical Character Recognition

PCA - Principal Component Analysis

QP - Quadratic Programming

RPT - Random Projection Technique

SDT - Shape Derivative Tool

SVMTC - SVM based training and classification

WWW - World Wide Web

Chapter 1

Introduction

1.1 Optical character recognition system

OCR system is a software which recognizes text in a document image. It does document image analysis in order to recognize a piece of text. Some of the applications of OCR system is discussed in the last chapter. It involves the area of image processing and pattern recognition. It also proves to be a great aid to the machine translation system by capturing the raw corpus from multilingual books[13]. It can also extract text from videos by doing document image analysis on the relevant frames in the video.

1.2 Challenges

The main challenge lies in tackling touching characters and recognizing characters which are almost similar.

1.3 Tamil Character Set

Tamil is a Dravidian language spoken in Tamil Nadu, India, North eastern Sri Lanka, Singapore and Malaysia. The language has 31 basic alphabets (12 vowels, 18 consonants and a special consonant) and the written script is comprises of 247 characters. However for character recognition there are only 155 classes which can be refered in A.1 including grantha characters. But this number excludes Tamil numerals which are uncommon.

1.4 Previous work

Previously Dr V. Krishnamoorthy, proposed a OCR system for Tamil language. The author claims 99% accuracy on printed text[1]. Later, Anbumani Subramanian and Bhadri Kubendran et.al proposed a system[2]. Again, it also claims better result. Both of the system doesn't perform page layout analysis and the system is no longer maintained. These systems uses moment based feature extraction technique.

In 2005, [3] proposed a system which did a comparative analysis of ANN and SVM for learning and classification. The SVM based classifier claims an accuracy of 66% for single font. They used the following visual features which is based on character glyph.

- Height of the character
- Width of the character
- Numbers of horizontal lines
- Numbers of vertical lines
- Numbers of circles
- Numbers of horizontally oriented arcs
- Numbers of vertically oriented arcs
- Centroid of the image
- Position of the various features
- Pixels in the various regions

In 2008,[4] proposed a OCR system which augments the power of ANN, SVM and HMM by fusing all of these together. In 2009, [5] did a comparative study of same recognition method. This system again uses character glyph based features. However, the author discussed recognition results only for selected number of character classes and the accuracy varies between 88% to 99%.

Chapter 2

System Architecture

2.1 Chapter Organisation

Preprocessing is discussed in the next chapter. Feature extraction, classification and recognition are discussed in the successive chapters. Finally we discuss the results and tools used along with our implementation of recognition engine based on supervised and unsupervised learning.

2.2 Architecture

System architecture shows the steps involved in the analysis of document image. The architecture of OCR system comprises of six stages and is shown in 2.1.

2.3 Deliverables

2.3.1 RPT

- Ground truth generating tool - train.py
- Blob extraction and preprocessing - createb.py
- Recognition engine - hyperplane2.py - our unsupervised classifier
- Result analyzer - confuse.sh

2.3.2 SVMTC

- Feature extractor - roger.m

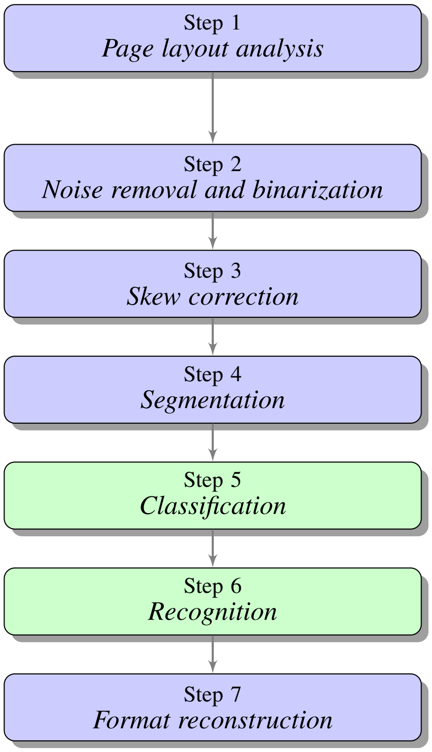


Figure 2.1: Architecture of a typical OCR system

Chapter 3

Preprocessing

3.1 Gray scale conversion

In grayscale conversion, 24-bit image is converted into 8-bit representation The input image to be recognised is converted into grayscale using the following formula.

$$I = 0.3R + 0.59G + 0.11B \tag{3.1}$$

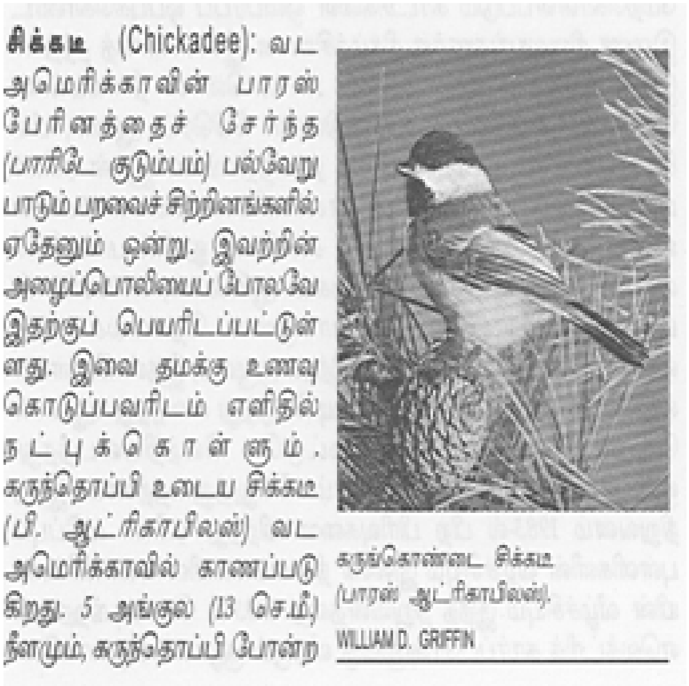
where R,G,and B corresponds to red, green and blue intensities respectively.

3.2 Thresholding

Thresholding refers to the pixel intensity to be used for binarization. For example, if threshold value is 120, all the pixels having value less than 120 are made zero and the rest as one for binary scale conversion.

3.3 Image Binarization

A binary image is a 2-bit representation of an image. Generally Otsu's thresholding algorithm or sauvola algorithm is used to binarize an image. But it all depends on the document.



சிக்கடீ (Chickadee): வட அமெரிக்காவின் பாரஸ் பேரினத்தைச் சேர்ந்த (பாரிடே குடும்பம்) பல்வேறு பாடும் பறவைச் சிற்றினங்களில் ஏதேனும் ஒன்று. இவற்றின் அழைப்பொலியைப் போலவே இதற்குப் பெயரிடப்பட்டுள்ளது. இவை தமக்கு உணவு கொடுப்பவரிடம் எளிதில் நட்புக்கொள்ளும். கருந்தொப்பி உடைய சிக்கடீ (பி. ஆட்ரிகாபிலஸ்) வட அமெரிக்காவில் காணப்படுகிறது. 5 அங்குல (13 செ.மீ) நீளமும், கருந்தொப்பி போன்ற

கருங்கொண்டை சிக்கடீ
(பாரஸ் ஆட்ரிகாபிலஸ்).
WILLIAM D. GRIFFIN

Figure 3.1: Sample image that is fed to tesseract

Figure 3.2: Sample gray scale blob

Figure 3.3: Sample binary scale blob

Figure 3.4: Sample of improper segmentation

3.4 Skew Detection and Correction

Generally, scanned document has to be skew-corrected because of incorrect orientation of document while scanning. There are various algorithms which can be used for skew correction. Some of the techniques as given in [13] are as follows: projection profile technique , Linear Regression analysis , Fourier Transform based method , nearest neighbor chain , Edge based connected component approach, interline cross-correlation, Entropy based methods. Jonathan J. H. presented a broad survey of existing techniques for skew correction.

3.5 Character segmentation

Character segmentation refers to chopping the words to individual blobs or characters. It can be carried out by performing connected component analysis[13] or ACM based method. Before carrying out character segmentation, text level and word level segmentation has to be done. Generally, the former is done by projection profile technique and the later by k-means clustering[13].

3.6 Page Layout Analysis

In page layout analysis, text region is separated from the non-text region and processed. Generally, voronoi segmentation or RAST based algorithm[14] is used for layout analysis.

3.7 Size Normalization

Each segmented character is normalized to fit within suitable matrix like 32 x 32 or 64 x 64 so that all characters are of same size .

3.8 Implementation

We make use of tesseract engine for carrying out the above listed steps. Finally it gives the bounding box information from which we extract the corresponding blobs.

Chapter 4

Feature Extraction

4.1 Active contour model

Active contour model[15] is a method to extract the contour of an image. It is also called as snakes model in literature. It is basically a spline defined by a set of points, internal energy and external energy. A snake initialized with a level set function refines iteratively and it is attracted towards the salient contour. It is generally based on the minimization of the energy. The snake is represented by n points as $v_i = (x_i, y_i)$ where i = 0,1..., n-1. The external energy of the snakes is given by

$$E_{external} = E_{image} + E_{con} \tag{4.1}$$

where $E_{external}$ refers to external energy, E_{image} refers to image force acting on the spline, E_{con} refers to the external constrained forces. The internal energy of the snakes is given by

$$E_{internal} = E_{cont} + E_{curv} \tag{4.2}$$

where $E_{internal}$ represents the internal energy of the spline (snake) due to bending, E_{cont} denotes the energy of the snake contour and E_{curv} denotes the energy of the spline curvature. Further E_{image} has got three components namely

- Lines
- Edges
- Terminations

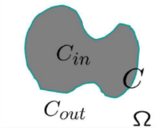


Figure 4.1: Evolution of contour, courtesy of Bresson

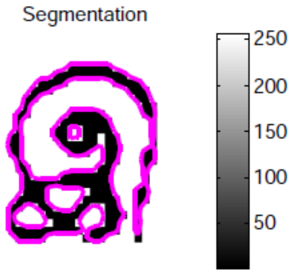


Figure 4.2: Segmentation using ACM

The above steps are the general description of ACM. In bresson's method[7], energy is minimized with SDT and calculus of variation. In order to minimize the energy, the problem is formulated as a convex optimization problem. It makes use of the level set method to minimize the gradient flow which in turn corresponds to the minimization of the energy functional. Since the level set minimization problem is a non-convex minimization problem, it is convexified by introducing dirac delta function. As the level set slowly evolves, finally we get a minimized energy contour.

In our case, ACM is used as a visual feature. The length of the maximum length contour, number of points in the contour matrix and number of rings are used as a feature.

4.2 Random Projection

Random Projection is a common dimensionality reduction technique[8]. The classes are classified based on L2-norm. We compare this technique with SVMTC. We also use RPT technique as feature for SVMTC. It is similar to PCA. The main difference is the introduction of random matrix.

4.3 Character geometry

Character geometry can serve as handy visual feature[9].

- Euler Number - It is defined as the difference of number of objects and number of holes in a image.
- Regional area - It is defined as the ratio of the number of the pixels in the skeleton to the total number of pixels in the image.
- Eccentricity - It is defined as the eccentricity of the smallest ellipse that fits the skeleton of the image.
- Zonal Feature - It is a feature based on the directional rules.

4.4 Hu's Moments

Image moment is a weighted average of pixel intensities. Properties that are derived from image moments include area and centroid. Information about image orientation can be obtained by taking second order moments. Scale invariant moments η_{ij} are obtained by dividing properly scaled μ_{00}th moment as follows.

$$\eta_{ij} = \frac{\mu_{ij}}{\mu_{00}{}^{(1+\frac{i+j}{2})}} \tag{4.3}$$

where i + j ≥ 2. The $(p+q)$th order of moment is geometric moment M_{pq} of a gray-level image is defined as

$$M_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^p y^p f(x,y)dxdy \tag{4.4}$$

In the case of a digital image, the double integral of the above equation must be replaced by a summation[11].

$$m_{pq} = \sum_{i=1}^{n}\sum_{j=1}^{n} i^p j^q f_{ij} \tag{4.5}$$

where N is the size of the image and f_{ij} are the grey levels of individual pixels.

4.5 Affine moment invariants

Affine moment invariants are image moments which is immune to affine transformation of image like scaling, translation and rotation. In this case, we use Hu's set of invariant moments[12].

4.6 Gabor Filter

The 2D-Gabor filter can serve as a statistical feature. It uses Gaussian kernel function. We create a vector of this function values with different frequencies and orientation.The standard deviation and mean of this vector serves as a feature.

$$f(x,y,\omega,\theta,\sigma_x,\sigma_y) = \frac{1}{2\pi\sigma_x\sigma_y} e^{[\frac{-1}{2}((\frac{x}{\sigma_x})^2+(\frac{y}{\sigma_y})^2)+j\omega(xcos\theta+ysin\theta)]} \tag{4.6}$$

4.7 Implementation

The feature extraction is carried out by roger.m. We extracted seven features based on Hu's Moments. Hundred based on RPT. Three based on ACM. Three hundred based on 2-D Gabor filter and the rest based on moment's invariants and character geometry.

Chapter 5

Training and classification

5.1 Support Vector Machine

Support Vector Machine[16] is machine learning technique. It is a form of supervised learning. It is used for classifying an object and for regression analysis. It takes a point in R^n and identifies to which of the two classes this point belongs. Initially a model is presented to SVM classifier. The model dictates to which class a sample input belongs. In technical parlance, SVM classifier constructs a hyperplane to separate the classes and later labels the new point to one of the classes and this type of classifier is called linear classifier. The maximum-margin hyperplane is the hyperplane which maximizes the distance between points of two different classes. Let D denote the training data and a set of points of the form

$$D = \{(x_i, y_i) | x_i \in R^p, y \in \{-1, 1\}\}^n \tag{5.1}$$

where y_i indicates whether the point x_i belongs to class -1 or 1. Our objective is find the maximum-margin hyperplane which separates class -1 from class 1. The sample on the margin is refered as support vector. Given

$$\mathbf{w} \cdot \mathbf{x} - b = 1 \tag{5.2}$$

and

$$\mathbf{w} \cdot \mathbf{x} - b = -1. \tag{5.3}$$

our objective is to choose w and b such that distance between the hyperplanes which separate one class from the other is maximized. The distance between the hyperplanes is given by $\frac{2}{\|w\|}$. So we want to maximize $\|w\|$. To prevent outliers we introduce the following constrains.

$$\mathbf{w} \cdot \mathbf{x}_i - b \geq 1 \qquad \text{for } \mathbf{x}_i \tag{5.4}$$

of first class

$$\mathbf{w} \cdot \mathbf{x}_i - b \leq -1 \qquad \text{for } \mathbf{x}_i \tag{5.5}$$

of the second class and which is same as

$$y_i(\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1, \quad \text{for all } 1 \leq i \leq n \tag{5.6}$$

which in turn can be formulated as follows.

Minimize $\|w\|$ subject to

$$y_i(\mathbf{w} \cdot \mathbf{x_i} - b) \geq 1 \tag{5.7}$$

5.1.1 Primal form

It is difficult to solve the preceding optimization problem. It is because it involves L2-norm which is computationally intensive. But it can be altered as $\frac{1}{2}\|\mathbf{w}\|^2$ by replacing $\|w\|$ without changing the solution. This is Quadratic programming optimization problem. More formally, Minimize

$$\frac{1}{2}\|\mathbf{w}\|^2 \tag{5.8}$$

subject to

$$y_i(\mathbf{w} \cdot \mathbf{x_i} - b) \geq 1 \tag{5.9}$$

The above constraint can be expressed as follows

$$\min_{\mathbf{w},b} \max_{\alpha} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i [y_i(\mathbf{w} \cdot \mathbf{x_i} - b) - 1] \right\} \tag{5.10}$$

Here we are locating the saddle point. Now solution for the above QP is given by,

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x_i} \tag{5.11}$$

and

$$b = \frac{1}{N_{SV}} \sum_{i=1}^{N_{SV}} (\mathbf{w} \cdot \mathbf{x_i} - y_i) \tag{5.12}$$

where N_{sv} is the number of support vectors.

5.1.2 Dual form

The objective in the dual form reveals that classification task is only a function of support vectors. After substituting for $||w||$, the SVM optimization problem becomes as follows. Maximize α_i subject to

$$\tilde{L}(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) \tag{5.13}$$

$\alpha_i \geq 0,$

Note that the kernel function here is defined as $k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$.

5.1.3 Soft margin

If there exists no hyperplane to separate two classes, then *soft margin* introduces slack variable ξ_i which measures degree of misclassification of the datum x_i as follows.

$$y_i(wx_i - b) \geq 1 - \xi_i, 1 \leq i \leq n \tag{5.14}$$

Now the maximizing objective function penalizes the non-zero ξ_i which is a trade-off between a large margin between hyperplanes and the error introduced. For linear penalty function, the optimization problem becomes as

$$\min_{\mathbf{w},\xi,b} \left\{ \frac{1}{2} ||\mathbf{w}||^2 + C \sum_{i=1}^{n} \xi_i \right\} \tag{5.15}$$

subject to

$$y_i(\mathbf{w}\cdot\mathbf{x_i}-b)\geq 1-\xi_i, \quad \xi_i\geq 0. \tag{5.16}$$

The above constraint with the objective of minimizing $\|w\|$ can be solved by using Lagrange multipliers as follows.

$$\min_{\mathbf{w},\xi,b}\max_{\alpha,\beta}\left\{\frac{1}{2}\|\mathbf{w}\|^2+C\sum_{i=1}^{n}\xi_i-\sum_{i=1}^{n}\alpha_i[y_i(\mathbf{w}\cdot\mathbf{x_i}-b)-1+\xi_i]-\sum_{i=1}^{n}\beta_i\xi_i\right\} \tag{5.17}$$

with $\alpha_i,\beta_i\geq 0$

5.1.4 Non-linear classification

Non-linear classification makes SVM, an extension of perceptron. Originally vapnik proposed the linear classifier and later it was extended for non-linear classification by applying kernel trick to maximum-margin hyperplane. In this case, dot product is replaced by non-linear kernel function. This makes it possible to represent the maximum-margin hyperplane in a transformed feature space. If the kernel function is Gaussian radial basis function, the corresponding feature space is infinite dimensional Hilbert space. Some common kernel functions used are as follows.

- Gaussian Radial Basis Function
- Hyperbolic tangent
- Polynomial (homogeneous)
- Polynomial (inhomogeneous)

SVM classifiers can be considered as form of Tikhonov regularization. Its special property is reducing error and increasing the margin of separation of different classes.

The SVM classifier is realized by solving the optimization problem formulated above. Generally Platts's Sequential Minimal Optimization (SMO) algorithm is used. It breaks down the problem into 2D subproblem. The other approach is based on interior point method which makes use of Newton iteration to solve Kuhn-Tucker conditions of primal and dual problem. But this method solves the problem as a whole. To avoid solving large kernel matrix, the matrix is rank approximated.

5.1.5 Multiclass SVM

Multiclass SVM aims to label more than two classes. Generally multiclass problem is reduced to binary classification problem. It is based on

- one-versus-all
- one-versus-one

Classification of new instances for one-versus-all case is done by a winner-takes-all strategy, in which the classifier with the highest output function assigns the class. For the one-versus-one approach, classification is done by a max-wins voting strategy, in which every classifier assigns the instance to one of the two classes, then the vote for the assigned class is increased by one vote, and finally the class with most votes determines the instance classification.

5.1.6 Parameter selection

Parameter selection is important because the effective of classifier is dictated by the kernel and its parameter(γ) and the soft margin(C) Parameter selection is carried out by performing grid-search with exponentially growing sequences of C and γ. Choices of parameter is checked by cross validation.

5.2 Implementation

The extracted feature vector is normalized and labelled. It is then passed to the svm learner for generation of a model. Finally the svm classifier is used to classify the new character based on the model that is created. We used a multiclass svm called SVM_struct. We took at-least ten samples from each class. Totally three classes were taken. SVM_struct

S.No	Class	Confusing class
1	tu	rru
2	vi	li
3	va	ya
4	wa	ta
5	va	na
6	va	la
7	ku	cu
8	mu	zhu
9	ku	ru
10	pa	pu
11	wi	rri
12	ki	ci
13	ka	ca

Table 5.1: Samples of least confusing class

is available at `http://download.joachims.org/svm_multiclass/examples/`. However, SVM_struct cannot take advantage of the core. So we used our feature extraction technique with MSVM[6] and it significantly reduced training time. In the case of MSVM, the trainmsvm tool creates a model based on the extracted features. Later, based on the model, predsvm tool classifies the blobs.

5.3 Hierarchal classification

We inspected the use of hierarchal classification for improving the accuracy of recognition. The discriminating feature is number of points of the contour. We selected the least confusing classes. Some of them are as follows.

Class	**Minimum**	**Maximum**
ku	263	294
ru	258	280
tu	256	282
rru	243	262
vi	227	288
li	244	259
ya	186	210
na	239	287
ta	246	256
va	245	264
wa	212	231
la	239	272
ku	263	294
cu	204	224
mu	228	267
zhu	244	286
ru	258	280
pa	146	166
pu	149	189
wi	249	268
rri	194	268
ki	270	300
ci	248	276
ka	212	225
ca	146	200

Table 5.2: Min and max no. of contour points for the least confusing classes

Chapter 6

Results

6.1 Dataset

The dataset is made up of 16 pages from a Tamil encyclopedia. However, we considered only two pages out of it for the whole experiment and also we segregated intractable characters from the rest. One page is used for creating the model and other page is used for testing.

6.2 Tools

6.2.1 Tesseract OCR Architecture

Tesseract has basically uses two import modules for character recognition.

- Character Chopper.
- Character Associator.

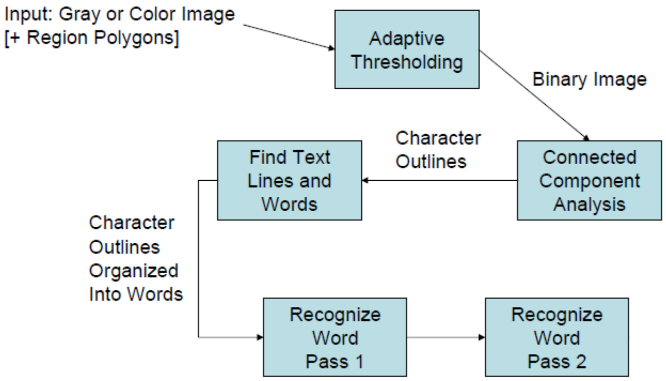

Input: Gray or Color Image
[+ Region Polygons]
Adaptive Thresholding
Binary Image
Connected Component Analysis
Character Outlines
Find Text Lines and Words
Character Outlines Organized Into Words
Recognize Word Pass 1
Recognize Word Pass 2

Figure 6.1: Architecture of tesseract engine, courtesy of Ray Smith

But additionally it makes use of Static classifier, Adaptive classifier and Dictionary for recognition.

6.2.2 Module description for making changes

- api – the plain api's that tesseract avails for creating custom OCR like ocropus gets into this directory.
- ccstruct – the common data structure that tesseract uses gets into this directory.
- classify – this directory contains modules for character classification.
- cutil – this directory contains file handling and heap management routines.
- dict – this directory contains word sense disambiguation routines for error correction and proper classification.
- tessdata – this is ideally the place where training data resides before it gets into /usr/share/tessdata or /usr/local/share/tessdata depending upon your installation configuration option.
- textord – this directory contains modules for preprocessing.
- wordres – this directory contains modules for recognition of characters.
- cmain – this directory contains essential modules for a fully functional OCR.
- java – this directory contains the piccolo client stub for interactive mode. It makes uses of the piccolo package for displaying word segmentation and for editing box files. It is available at `http://www.cs.umd.edu/hcil/jazz/team/index.shtml`.Make sure that piccolox.jar is in the classpath.

Classpath: bash> export CLASSPATH=$CLASSPATH:/path/to/my/piccolo.jar

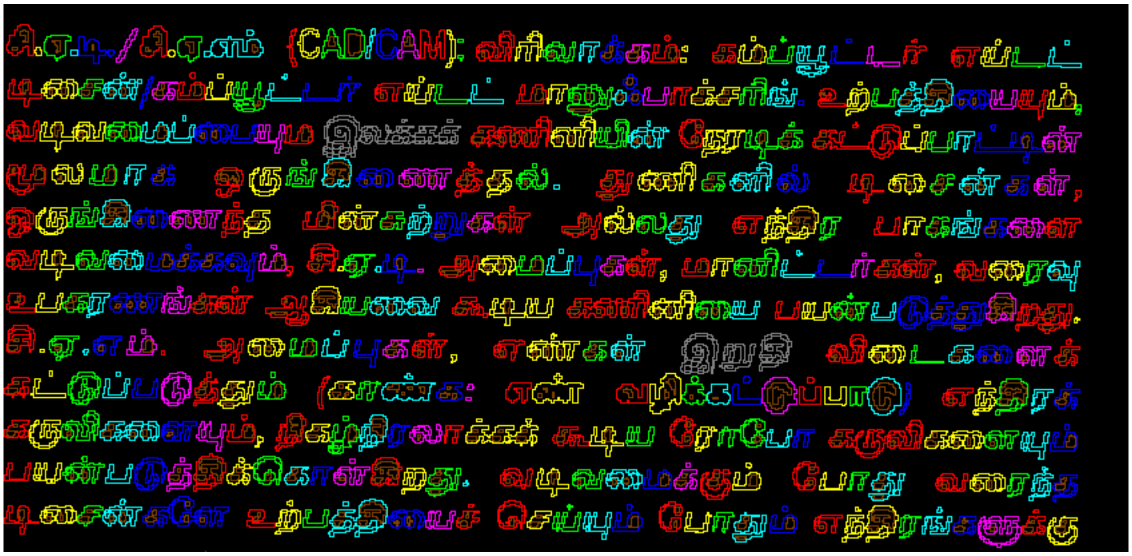


Figure 6.2: Tesseract engine in interactive mode

6.2.3 Classification

It does Polygonal approximation of each image to get rid of noise and extracts feature from it for classification. It uses static classifier for classifying broken characters and it can be time consuming.

6.2.4 Features

It can handle unicode and it already supports bunch of languages.

6.2.5 Running Tesseract

We can run Tesseract either in Normal or Interactive mode. In the interactive mode, the user could visualize the chopped blobs and if needed could edit the bounding box of each of the characters in the document image.

Normal mode: bash> tesseract sam.tif out [-l eng]

Interactive mode: bash> tesseract sam.tif out [-l eng] inter

In the above commands ’eng’ stands for English. It is the language code. By default tesseract uses English as the recognition language. However, it doesn’t matter in our case

Junk	Top	Left	Right	Bottom	Page no
@	865	2893	883	2913	0
Q	867	2875	878	2885	0
w	879	2875	889	2885	0
w	867	2852	881	2862	0

Table 6.1: Sample bounding box information for four blobs in a page

because we basically use it only for extracting the bounding box. The output is written to out.txt in this case.

6.2.6 Extracting bounding box information

If you want Tesseract to do only preprocessing and get the blob information, you could easily do that as follows.

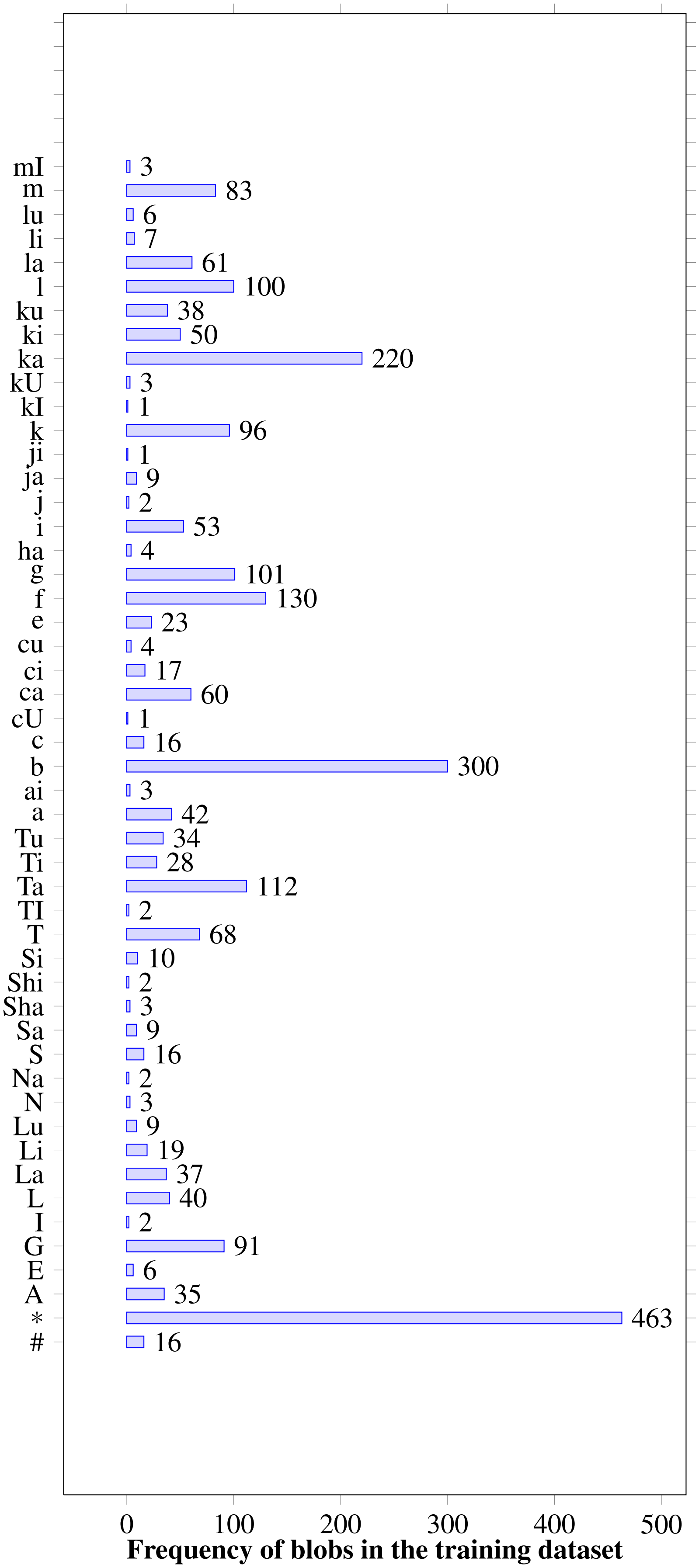
- bash> tesseract sam.tif out makebox

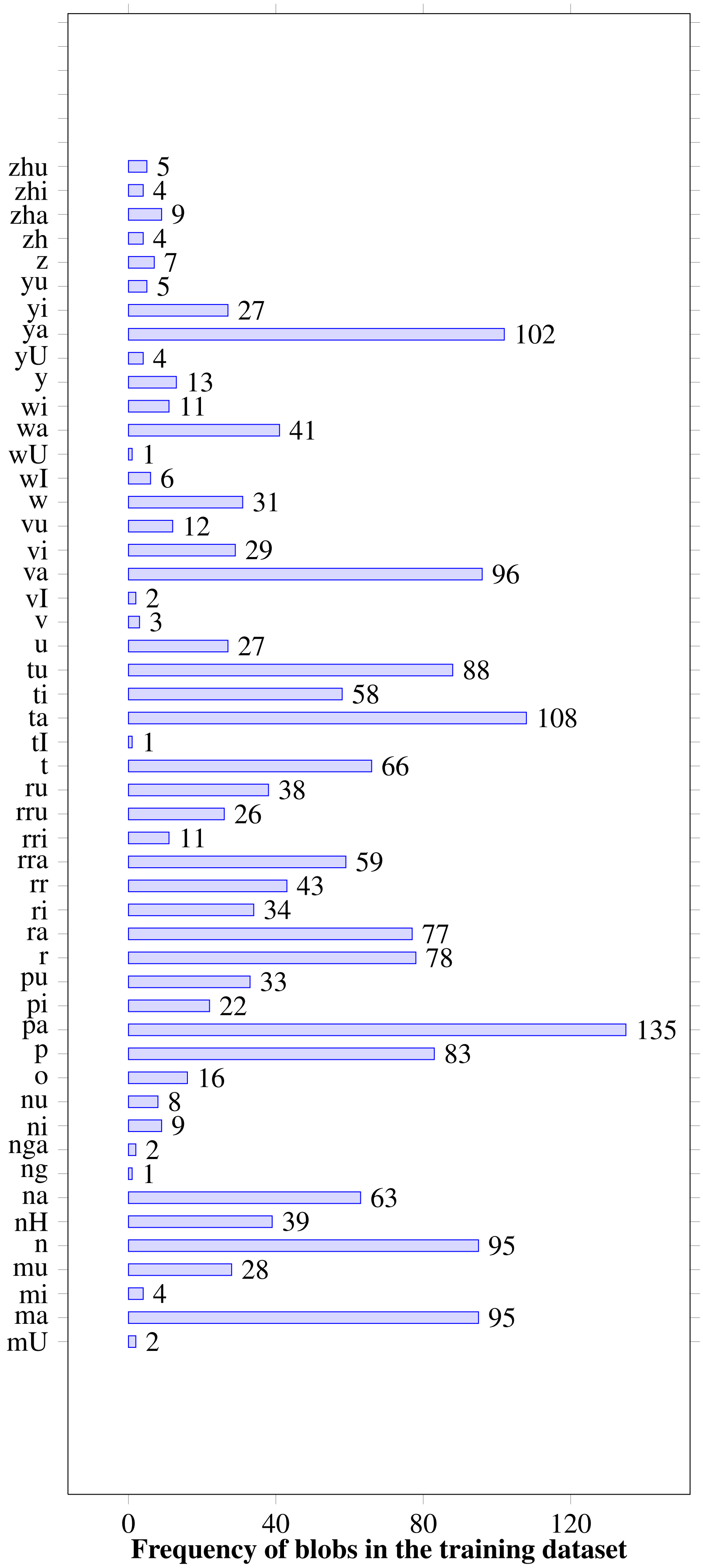
The blob information is written into out.box. After this you could use our blob extractor to get the blobs to be used for recognition.

6.2.7 Using Moshpytt with Tesseract

As mentioned earlier, Moshpytt could greatly reduce our work of creating training samples for new languages with tesseract. The tool can also be used to edit the bounding box related information. It is written in python and we use it only to edit bounding box information since we are using our oown method based on RPT and SVMTC for character recognition. It can be invoked by running

```
bash> python moshpytt.py #editing box file
```


mI 3
m 83
lu 6
li 7
la 61
l 100
ku 38
ki 50
ka 220
kU 3
kI 1
k 96
ji 1
ja 9
j 2
i 53
ha 4
g 101
f 130
e 23
cu 4
ci 17
ca 60
cU 1
c 16
b 300
ai 3
a 42
Tu 34
Ti 28
Ta 112
TI 2
T 68
Si 10
Shi 2
Sha 3
Sa 9
S 16
Na 2
N 3
Lu 9
Li 19
La 37
L 40
I 2
G 91
E 6
A 35
* 463
# 16
0
100
200
300
400
500
Frequency of blobs in the training dataset


zhu 5
zhi 4
zha 9
zh 4
z 7
yu 5
yi 27
ya 102
yU 4
y 13
wi 11
wa 41
wU 1
wI 6
w 31
vu 12
vi 29
va 96
vI 2
v 3
u 27
tu 88
ti 58
ta 108
tI 1
t 66
ru 38
rru 26
rri 11
rra 59
rr 43
ri 34
ra 77
r 78
pu 33
pi 22
pa 135
p 83
o 16
nu 8
ni 9
nga 2
ng 1
na 63
nH 39
n 95
mu 28
mi 4
ma 95
mU 2
0
40
80
120
Frequency of blobs in the training dataset

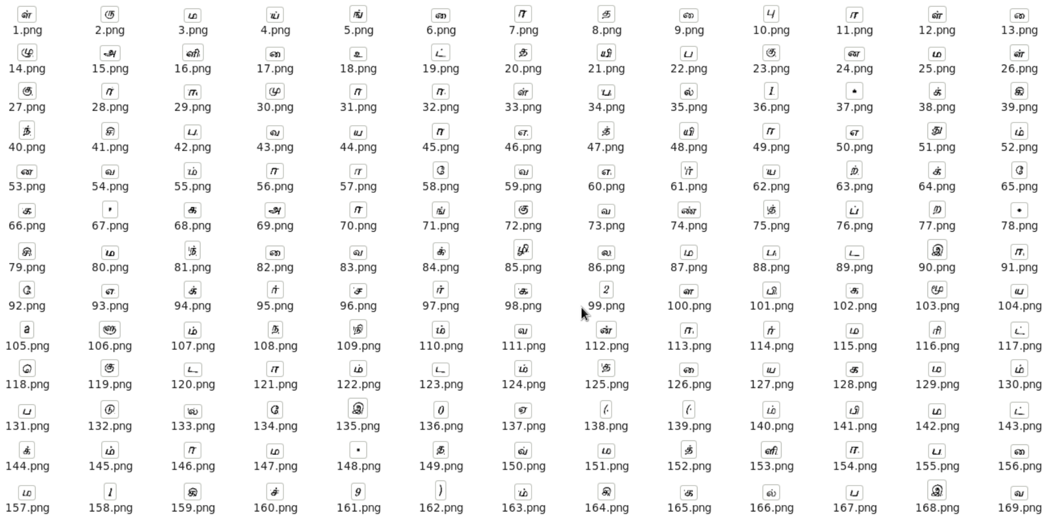


Figure 6.3: Sample output after preprocessing

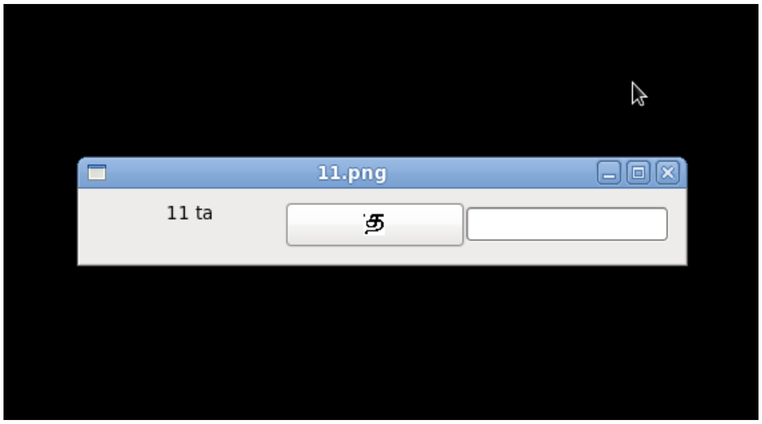


Figure 6.4: Ground truth generation

6.2.8 Custom tools

As part of the project, a blob extractor was developed. It can extract the blob based on the bounding box information given by tesseract. The ground truth was generated with the help of an handy interface developed in PyGTK.

6.2.9 Elusive samples

We inspected sample blobs for hierarchal classification. Here are some of the elusive cases It is called so because noise or improper segmentation makes them stray blobs. Normal contour for class ’ya’ is one. But in the figure 6.7 it is three.

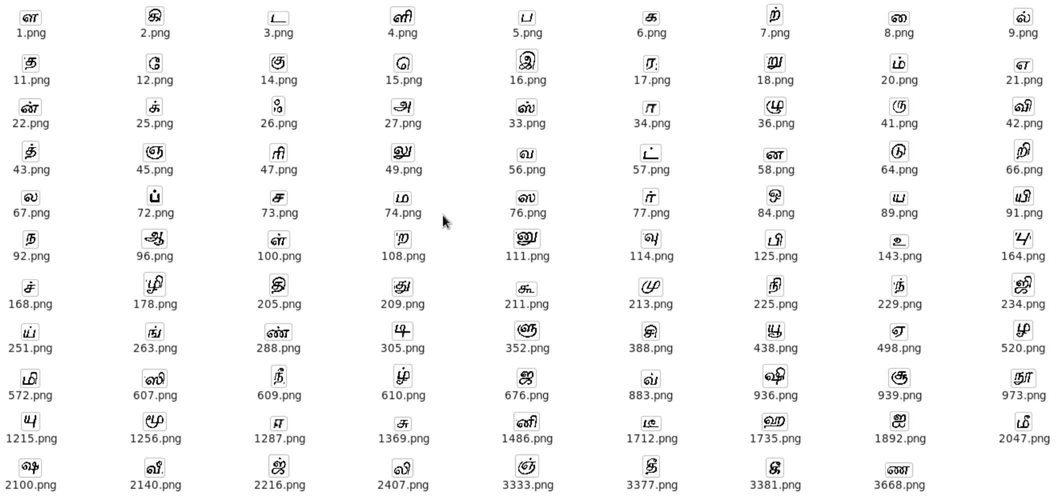


Figure 6.5: Model presented to the recognition engine

			Samples		Accuracy	
S.No	Method	Features	Training	Testing	Training	Testing
1*	SVMTC	555	4201	4067	75	< 44 (3 classes)
2**	RPT	1000	155 classes	4201	98	59

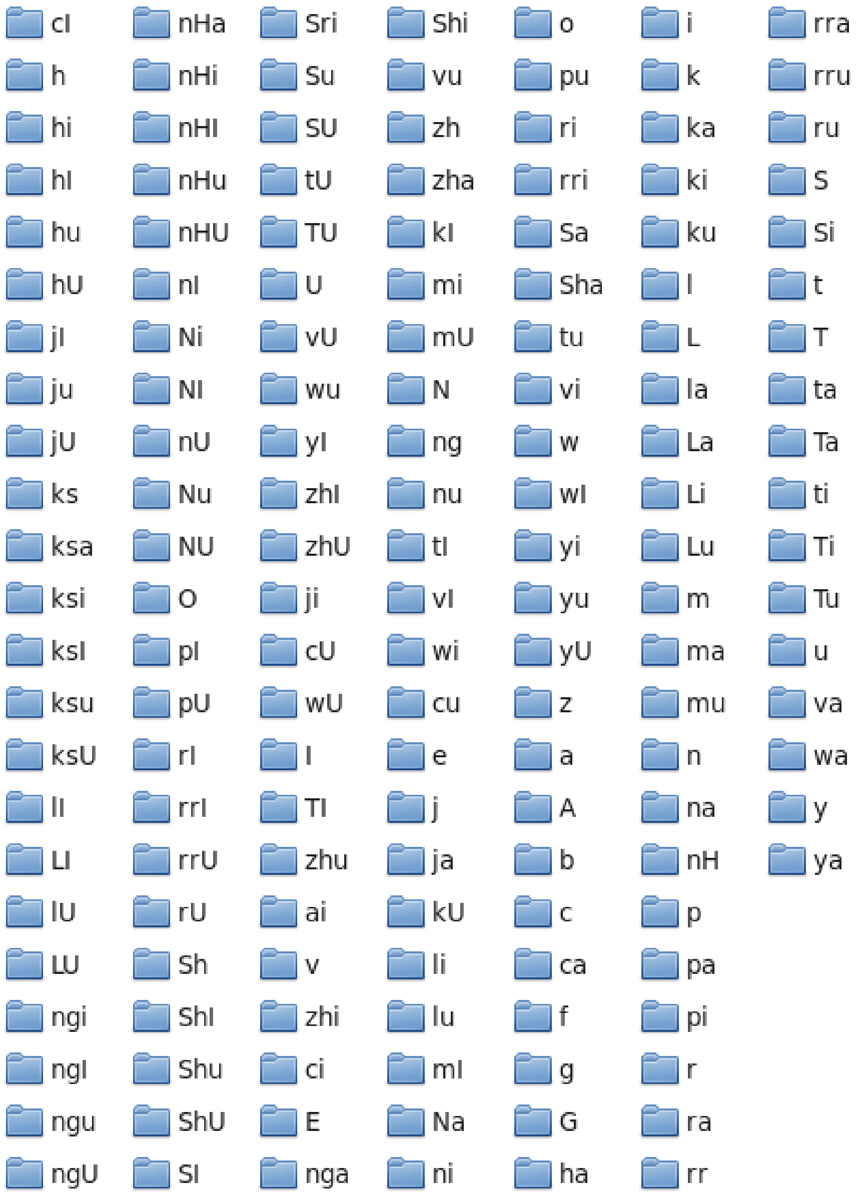
Table 6.2: Recognition result based on SVMTC and RPT

6.2.10 Confusion matrix

The confusion matrix is generated with the help of ground truth and some handy shell scripts.

6.2.11 Statistics

The results are based on training and testing dataset generated as part of this project. It is a comparison of supervised and unsupervised technique for character recognition. It should be noted that tesseract doesn't perform perfect segmentation as expected because of touching characters. So junk and improper segmentation of blobs are removed from training and testing dataset. However, there are some stray characters in the dataset due to the reason stated above. So the accuracy is calculated excluding the stray characters.



cI
h
hi
hI
hu
hU
jI
ju
jU
ks
ksa
ksi
ksI
ksu
ksU
lI
LI
lU
LU
ngi
ngI
ngu
ngU
nHa
nHi
nHI
nHu
nHU
nI
Ni
NI
nU
Nu
NU
O
pI
pU
rI
rrI
rrU
rU
Sh
ShI
Shu
ShU
SI
Sri
Su
SU
tU
TU
U
vU
wu
yI
zhI
zhU
ji
cU
wU
I
TI
zhu
ai
v
zhi
ci
E
nga
Shi
vu
zh
zha
kI
mi
mU
N
ng
nu
tI
vI
wi
cu
e
j
ja
kU
li
lu
mI
Na
ni
o
pu
ri
rri
Sa
Sha
tu
vi
w
wI
yi
yu
yU
z
a
A
b
c
ca
f
g
G
ha
i
k
ka
ki
ku
l
L
la
La
Li
Lu
m
ma
mu
n
na
nH
p
pa
pi
r
ra
rr
rra
rru
ru
S
Si
t
T
ta
Ta
ti
Ti
Tu
u
va
wa
y
ya

Figure 6.6: Model generation uses one instance of each of these class

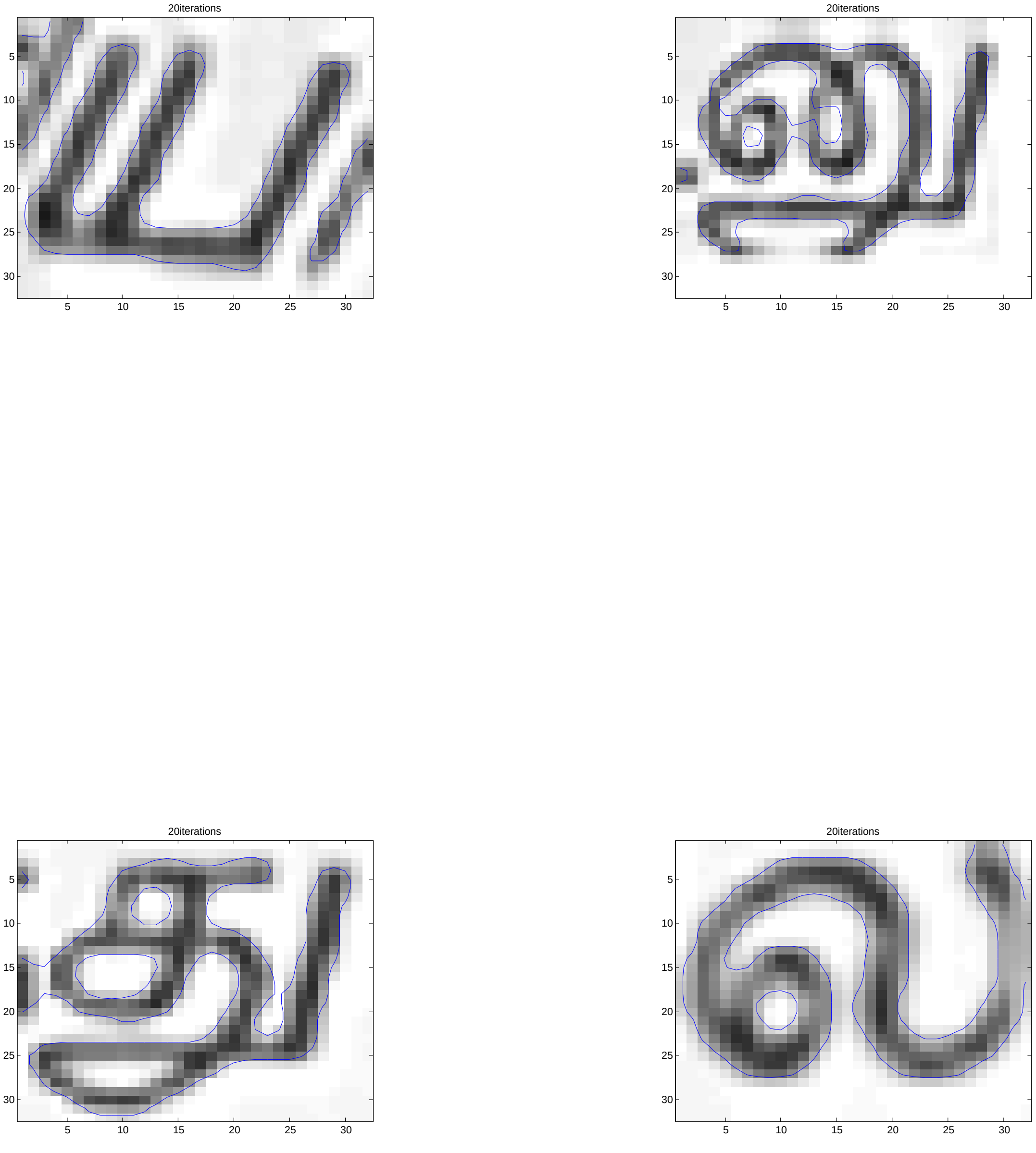


Figure 6.7: Some sample elusive contour from dataset

	ட	து	ந	ன	ன்	ப	பு	ப்	ம	ய	ற
ட	92					15					2
து		53				3					
ந			33			1					5
ன				55		1					
ன்	1				56	3					
ப	34					97	2				
பு						1	1				
ப்	2					5		1			
ம	2					2			80		
ய	1					16				61	
ற			2			2					47

Figure 6.8: Sample confusion matrix

Chapter 7

Application and Conclusion

7.1 Application

OCR system is useful for creating volume of corpus by converting existing scanned document into plain text. It can also be used for creating encyclopedia from existing old documents. Since multilingual encyclopedia can serve as parallel corpus, OCR system would be a handy tool for creating dataset from existing paperback encyclopedia. Much of the WWW document images would be useful only if it can be indexed because only then it becomes searchable. So OCR plays a major role there too. OCR system can also be used in post office where handwritten address is converted into digital format for efficient storage and retrieval. Apart from these examples, OCR system can be used anywhere when there is need to convert document image to text.

7.2 Conclusion and Future enhancement

In this work, we are presenting a comparative study of recognition based on supervised and unsupervised learning. The former is based on SVM based training and classification(SVMTC) and the later is based on Random Projection Technique(RPT).Overall, RPT gives better results than SVMTC. The accuracy of recognition can be improved based on hierarchal classification. In order to deal with similar characters, nearest neighbor search can be carried out. At the word level, the recognition can be improved by making use of a dictionary. Finally, all the associated scripts and datasets are made available at `https://github.com/pradeepmathesh/tocr` for future enhancement.

References

[1] V. Krishnamoorthy, ”OCR Software for Printed Tamil Text”, *Proceedings of Tamil Internet 2002*, pp. 99-101, 2002

[2] Anbumani Subramanian and Bhadri Kubendran, ”Optical Character Recognition of Printed Tamil Characters”,`http://www.angelfire.com/in/anbu/vt/tamilocr/`,(Accessed on: December 2011)

[3] Seethalakshmi et.al, ”Optical Character Recognition for printed Tamil text using Unicode”,*Journal of Zhejiang University Science*, pp. 1297, Sept.10, 2005

[4] Jagadeesh et.al,”Accuracy Augmentation of Tamil OCR Using Algorithm Fusion”,*International Journal of Computer Science and Network Security*, VOL.8 No.5, pp.51, May 2008

[5] Jagadeesh et.al,”A Comparative Study of Optical Character Recognition for Tamil Script”,*European Journal of Scientific Research*, Vol.35 No.4, pp.570-582, 2009

[6] F. Lauer, ”MSVMpack: a Multi-Class Support Vector Machine Package”, `http://www.loria.fr/~lauer/MSVMpack`,2011,(Accessed on: January 2012)

[7] Xavier Bresson, ”A Short Guide on a Fast Global Minimization Algorithm for Active Contour Models”, Preprint, April 22, 2009

[8] Qinfeng Shi, ”Rapid Face Recognition Using Hashing”, Preprint, 2009

[9] Dinesh Dileep, ”A feature extraction technique based on character geometry for character recognition”, `http://www.ece.iisc.ernet.in/~dileep`, 2008,(Accessed on: December 2011)

[10] Ray Smith, ”An Overview of the Tesseract OCR Engine”, *IEEE Trans. on Image Processing*, 2007

[11] Jan Flusser and Tomas Suk, ”On the Calculation of Image Moments”,*Pattern Recognition Letters*, January 1999

[12] Jan Flusser et.al,*Moments and Moment Invariants in Pattern Recognition*, 2009

[13] Praveen Krishnan, ”Preprocessing Algorithms for Tamil Optical Character Recognition”, Master’s Thesis, Amrita Vishwa Vidyapeetham, CEN, 2011

[14] Amy Alison Winder, ”Extending the page segmentation algorithms of the ocropus documentation layout analysis system”, Master’s Thesis, Boise state university, Department of CS, July 2010

[15] SVM - Wikipedia, the free encyclopedia, ”`http://en.wikipedia.org/wiki/SVM`”, (Accessed on: July 2011).

[16] Active contour model - Wikipedia, the free encyclopedia, ”`http://en.wikipedia.org/wiki/Active_contour_model`”, (Accessed on: July 2011).

Appendix A

Appendix

A.1 Character classes in Tamil OCR

The following list presents the character classes in Tamil OCR along with its transliterated mnemonic. Classes are transliterated for convenience.

Class	Label	Transliteration
அ	1	a
ஆ	2	A
இ	3	i
ஈ	4	I
உ	5	u
ஊ	6	U
எ	7	e
ஏ	8	E
ஐ	9	ai
ஒ	10	o
ஓ	11	O
க்	12	k
ங்	13	nH
ச்	14	c
ஞ்	15	ng
ட்	16	T
ண்	17	N
த்	18	t
ந்	19	w
ப்	20	p
ம்	21	m
ய்	22	y
ர்	23	r
ல்	24	l
வ்	25	v
ழ்	26	zh
ள்	27	L
ற்	28	rr
ன்	29	n
க	30	ka
ங	31	nHa
ச	32	ca
ஞ	33	nga
ட	34	Ta
ண	35	Na
த	36	ta
ந	37	wa
ப	38	pa
ம	39	ma
ய	40	ya
ர	41	ra
ல	42	la
வ	43	va
ழ	44	zha
ள	45	La
ற	46	rra
ன	47	na
கி	48	ki
ஙி	49	nHi
சி	50	ci
ஞி	51	ngi
டி	52	Ti

Class	Label	Transliteration
ணி	53	Ni
தி	54	ti
நி	55	wi
பி	56	pi
மி	57	mi
யி	58	yi
ரி	59	ri
லி	60	li
வி	61	vi
ழி	62	zhi
ளி	63	Li
றி	64	rri
னி	65	ni
கீ	66	kI
ஙீ	67	nHI
சீ	68	cI
ஞீ	69	ngI
டீ	70	TI
ணீ	71	NI
தீ	72	tI
நீ	73	wI
பீ	74	pI
மீ	75	mI
யீ	76	yI
ரீ	77	rI
லீ	78	lI
வீ	79	vI
ழீ	80	zhI
ளீ	81	LI
றீ	82	rrI
னீ	83	nI
கு	84	ku
ஙு	85	nHu
சு	86	cu
ஞு	87	ngu
டு	88	Tu
ணு	89	Nu
து	90	tu
நு	91	wu
பு	92	pu
மு	93	mu
யு	94	yu
ரு	95	ru
லு	96	lu
வு	97	vu
ழு	98	zhu
ளு	99	Lu
று	100	rru
னு	101	nu
கூ	102	kU
ஙூ	103	nHU

Class	Label	Transliteration
சூ	104	cU
ஙூ	105	ngU
டூ	106	TU
ணூ	107	NU
தூ	108	tU
நூ	109	wU
பூ	110	pU
மூ	111	mU
யூ	112	yU
ரூ	113	rU
லூ	114	lU
வூ	115	vU
ழூ	116	zhU
ளூ	117	LU
றூ	118	rrU
னூ	119	nU
◌ெ	120	g
◌ே	121	G
◌ை	122	f
ஸ்ரீ	123	Sri
ஜ	124	ja
ஜி	125	ji
ஜீ	126	jI
ஜு	127	ju
ஜூ	128	jU
ஜ்	129	j
ஷ	130	Sha
ஷி	131	Shi
ஷீ	132	ShI
ஷு	133	Shu
ஷூ	134	ShU
ஷ்	135	Sh
ஸ	136	Sa
ஸி	137	Si
ஸீ	138	SI
ஸு	139	Su
ஸூ	140	SU
ஸ்	141	S
ஹ	142	ha
ஹி	143	hi
ஹீ	144	hI
ஹு	145	hu
ஹூ	146	hU
ஹ்	147	h
க்ஷ	148	ksa
க்ஷி	149	ksi
க்ஷீ	150	ksI
க்ஷு	151	ksu
க்ஷூ	152	ksU
க்ஷ்	153	ks
◌∘∘	154	z
◌ா	155	b