Integration of Bangla script recognition support in OCRopus

A Thesis
Submitted to the Department of Computer Science and Engineering

of

BRAC University

By
Muttakinur Rahman Chowdhury (Shouro)

Supervisor
Dr. Mumit Khan

Co-supervisor
Md. Abul Hasnat

In Partial Fulfillment of the
Requirements for the Degree

of

Bachelor of Science in Computer Science
December 2008

DECLARATION

I hereby declare that this thesis is based on the results found by myself. Materials of work found by other researcher are mentioned by reference.

Signature of Supervisor

Signature of Author

Acknowledgement

Special thanks to Dr. Mumit Khan who gave me guidance through my thesis research work and also for his support in the whole process. And also show him my gratitude for considering me as to do the thesis under his supervision. I also thank Md. Abul Hasnat my co- Supervisor for his help a guidance.

I also like to thank

Mark Stillwell (University of Hawaii)
Prof. Thomas Breuel (IUPR)
Ilya Mezhirov (IUPR)
Yves Rangoni (IUPR)
Ray Smith (Google research)
CRBLP Lab

Abstract

OCRopus is an open source state-of-the-art document analysis and OCR system, featuring pluggable layout analysis, pluggable character recognition, statistical natural language modeling, and multi-lingual capabilities. The system is being developed with the generous support from Google and other organizations. One of the major goals of OCRopus is to make it multi-lingual. Researchers and developers from different languages involved them with this project to integrate their language or script support. The aim of this thesis is to integrate Bangla script recognition support in OCRopus. The major tasks of this thesis will be: experiment the existing algorithms of OCRopus and check their usability for Bangla script, learn the training and recognition procedures and finally integrate Bangla script related available algorithms within OCRopus to support the complete recognition.

INTRODUCTION **6**

OCROPUS-BPNET **8**

Training bpnet 8
Testing bpnet classifier 10
Best Neural Net Parameters for Training 10
Result Analysis and discussion 11

TESSERACT **12**

Tesseract installation: 12
How to train Bangla and Devanagari script for tesseract engine 12
Testing tesseract 16
Result Analysis and discussion 20

CRBLP SEGMENTER PLUGIN **21**

HOW TO ADD A METHOD IN OCROPUS-0.2 ..... **21**

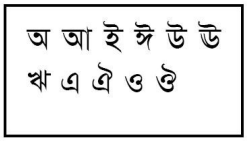
CONCLUSION **22**

REFERENCES **23**

Introduction

OCRopus is a state-of-the-art document analysis and OCR system, featuring pluggable layout analysis, pluggable character recognition, statistical natural language modeling, and multi-lingual capabilities. The system is being developed with the generous support from Google and other organizations; the primary developers are at the IUPR Research Group at the DFKI Research Center. Ocropus has two different recognition engines. One of them is bpnet classifier and the other one is tesseract. The Bpnet classifier is still in testing and development phase where Tesseract is completely matured. The Tesseract OCR engine was one of the top 3 engines in the 1995 UNLV Accuracy test. Between 1995 and 2006 it had little work done on it, but it is probably one of the most accurate open source OCR engines available.

In Bangla script there are 11 independent vowel and 39 consonant characters. These are called basic characters (Fig. 1). From Fig. 1 it is noted that most of the characters have a horizontal line at the upper part. This horizontal line is called headline or 'matraa' which is used to connect the characters in a word. Out of 50 basic characters 32 characters have this matra (head line). Out of these 11 independent vowels 10 vowels has dependent form (Fig. 2). The dependent vowels with examples are shown in Table 1. If the first character of the word is a vowel then it remains its independent form [4]. If the first character of the word is a vowel then it remains its independent form. Generally a vowel following by a consonant takes a dependent form and placed at the left or right or both or bottom of the consonant.



(a) Vowels

(b) Consonants

Figure 1: Basic characters of Bangla script

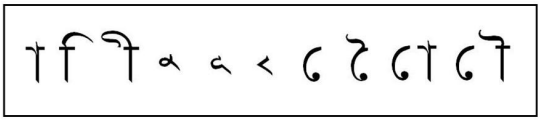


Figure 2: Dependent vowels

		Independent shape	Example	Dependent shape	Example
1.		আ	আমরা	া	কাল
2.		ই	ইদুর	ি	তিন
3.		ঈ	ঈদ	ী	নীট
4.		উ	উট	ু	তুমি
5.		ঊ	ঊষা	ূ	পূজা
6.		ঋ	ঋণ	ৃ	কৃষি
7.		এ	এক	ে	কেমন
8.		ঐ	ঐরাবত	ৈ	তৈল
9.		ও	ওল	ো	তোমরা
10.		ঔ	ঔষধ	ৌ	নৌকা

Table 1: Example of Independent and dependant vowels

In Bangla script, a consonant followed by another consonant sometime takes a compound shape which we call as compound character. Example of few compound characters is shown in Table 2. There is around 270 compound characters present in Bangla script [5], most of which are formed by consonant-consonant combination. Compounding of three consonants is also possible. Among the listed compound characters there are few characters which actually do not change their shape; instead they just attach a consonant modifier with them as like the vowel modifiers. In Bangla script there are 3 consonant modifiers which completely change their shape. These compound characters can also be followed by dependent vowels.

Consonant Combination	Compound character
ক ্ ক	ক্ক
ক ্ ত	ক্ত
ক ্ ন	ক্ন
ক ্ ম	ক্ম
ক ্ ষ	ক্ষ
ঙ ্ ক	ঙ্ক
ক ্ ল	ক্ল
ল ্ ক	ল্ক

Table 2: Example of compound characters

After a complete analysis on Ocropus we found that, to integrate Bangla script recognition support in Ocropus we need a complete training data as well as a complete character segmenter. The training data will be different for bpnet and tesseract.

OCRopus-bpnet

bpnet is a Multi Layer Perceptron Neural Network classifier.

Training bpnet

Two Approaches:
1. From individual character image.
2. From text line image.

1st approach:
In this approach individual unicode character image and its transcript is listed in a list file and using this list file training engine train bpnet.

2nd approach:
In this approach text image line, its transcript and its colored image are mapped together in a list file and then using this list file training engine train bpnet.

Recognition bpnet
Generate a color segmented character image.
Pass this image to the bpnet classifier.

Training data generation for bpnet

Approach 1: From individual character image

Figure: Character Image

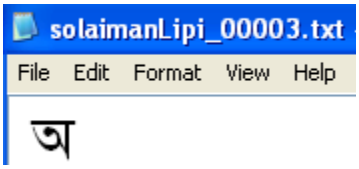


Figure: Character Unicode

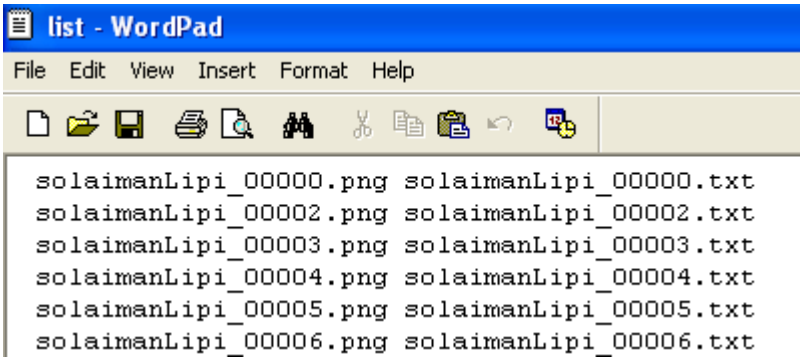


Figure - Image – transcription mapped list file

Approach 2: From line image

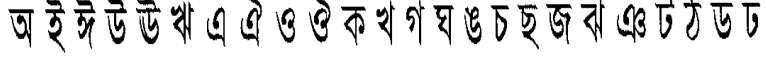


Figure: Line Image

Figure: Color Segmented Image

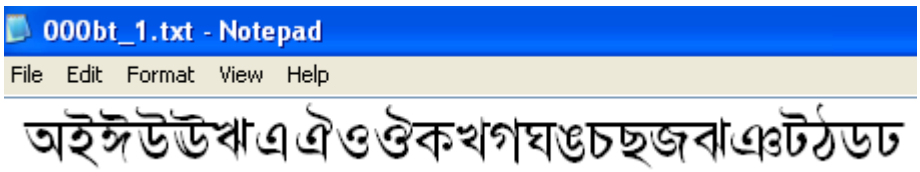


Figure: Transcription

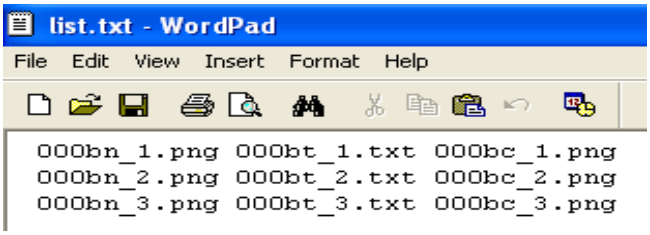


Figure: Image – transcription mapped list file

Testing bpnet classifier

Step 1: Generate a color segmented character image.

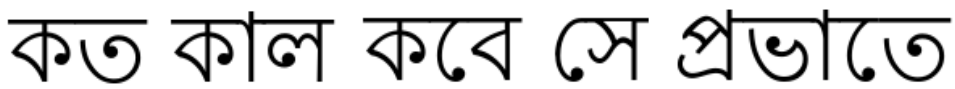
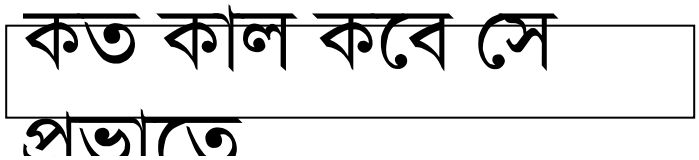


Figure: Input text Image

Figure: Segmented color Image

Step 2: Pass this image to the bpnet classifier.



hOCR output

Best Neural Net Parameters for Training

nhidden - 500
epochs - 300
learningrate - 0.2
testportion - 0
normalize - 1
shuffle – 1

Result Analysis and discussion

We perform the result testing on isolated characters as well as connected characters (words). For isolated characters the accuracy is 100% if we consider training and testing characters as similar font and size. For different font the accuracy goes down to 65%. For connected character recognition reported recognition accuracy is up to 70%. We did not perform any test on different font connected character recognition. Table 3 represents the results of out testing.

Type of image	**Font**	**Accuracy**
Isolated character	same	100%
Isolated character	different	60%
Word image	same	70%

Table 3: Experimental result

Tesseract

Tesseract installation:

extract tesseract package
get inside the tesseract directory
run these commands:
#./configure
#make
#make install

How to train Bangla and Devanagari script for tesseract engine

This document provides the step-by-step instructions that we followed to train data for Bangla and Devanagari script. This is just a short version of the document TrainingTesseract, which we followed to prepare training data for Bangla and Devanagari. No detail explanation for the purpose of each step is given here.

Data files required

To train Bangla or Devanagari scripts (lang = ban/dev), you have to create 8 data files in the tessdata subdirectory. The 8 files are:

1. tessdata/lang.freq-dawg

2. tessdata/lang.word-dawg

3. tessdata/lang.user-words

4. tessdata/lang.inttemp

5. tessdata/lang.normproto

6. tessdata/lang.pffmtable

7. tessdata/lang.unicharset

8. tessdata/lang.DangAmbigs

Step by step procedure

Step – 1: Create training data

Preparing training data depends on the characters or units that you want to recognize. Decision about the number of training data units depends on the performance of the segmentation algorithm. If you

consider minimal segmentation then you have to consider all the combinations formed by the alphabets of your script. However if your segmentation algorithm is well enough to segment the basic units properly then you can train only the basic and compound units of your script. In the fundamental level we consider training only the basic units. An example of the training data units is shown in figure-1 (Bangla training data) and figure-2 (Devanagari training data).

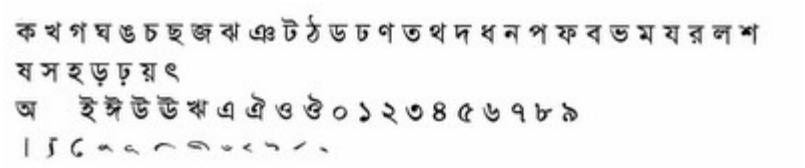


Figure 1: Training data units for Bangla script

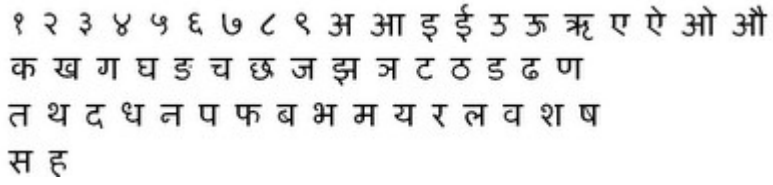


Figure 2: Training data units for Devanagari script

Step – 2: Make Box file

In this step we have to prepare a box file (a text file that lists the characters in the training image, in order, one per line, with the coordinates of the bounding box around the image). To create the box file, run Tesseract on each of your training images using this command line. The command is as follows:

tesseract trainfile.tif trainfile batch.nochop makebox

This will generate a file named trainfile.txt that you have to rename as trainfile.box. Then manually edit the box file where you have to replace the Latin characters (first character of each line) with appropriate unicode Bangla/Devanagari character. If any particular character is broken into two boxes then you have to manually merge the boxes. An example of edited box file is shown in figure-3. The generated box file name must be same with the training tif image file name.

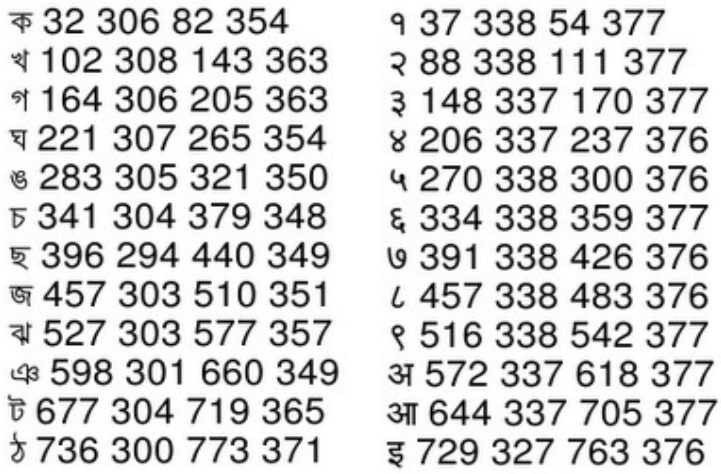


Figure 3: Box file for Bangla and Devanagari script

Step – 3: Run Tesseract for Training

For each of your training image and boxfile pairs, run Tesseract in training mode using the following command:

tesseract trainfile.tif junk nobatch box.train

This will generate a file named trainfile.tr which contains the features of each character of the training page.

Step – 4: Clustering

Clustering is necessary to create prototypes. The character shape features can be clustered using the mftraining and cntraining programs. The mftraining program is invoked using the following command:

mftraining trainfile.tr

This will output two data files: inttemp and pffmtable. (A third file called Microfeat is also written by this program, but it is not used.)

The cntraining program is invoked using the following command:

cntraining trainfile.tr

This will output the normproto data file.

In case of multiple training data the following command will be used:

mftraining trainfile_1.tr trainfile_2.tr ...

cntraining trainfile_1.tr trainfile_2.tr ...

Step – 5: Compute the Character Set

Next you have to generate the unicharset data file using the following command:

unicharset_extractor trainfile.box

This will generate a file named unicharset. Tesseract needs to have access to character properties isalpha, isdigit, isupper, islower. To set these properties we have to manually edit the unicharset file and change the default value (0) set for each training character. An example of the unicharset file is shown in figure-4.

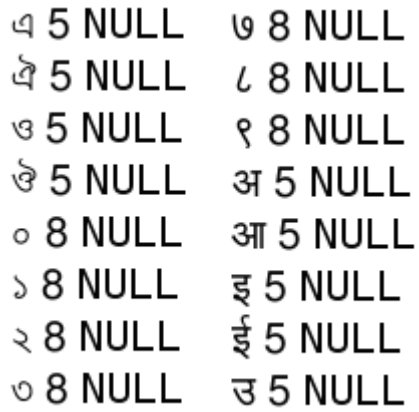


Figure 4: unicharset file for Bangla and Devanagari script

Step – 6: Prepare Dictionary data

Tesseract uses 3 dictionary files for each language. Two of the files are coded as a Directed Acyclic Word Graph (DAWG), and the other is a plain UTF-8 text file. To make the DAWG dictionary files, you first need a wordlist for your language. The wordlist is formatted as a UTF-8 text file with one word per line. Split the wordlist into two sets: the frequent words, and the rest of the words, and then use wordlist2dawg to make the DAWG files:

wordlist2dawg frequent_words_list freq-dawg

wordlist2dawg words_list word-dawg

The third dictionary file is called user-words and is usually empty.

The dictionary files freq-dawg and word-dawg don't have to be given many words if you don't have a wordlist to hand, but accuracy will be lower.

Step – 7: Prepare DangAmbigs file

This file represents the intrinsic ambiguity between characters or sets of characters. You have to generate this file considering the recognition failure example in your script. An example of the rules is shown in figure-5 for Bangla script.

Figure 5: Ambiguity between characters in Bangla script

Step – 8: Rename the necessary files

As mentioned in the starting of this document, now you have to rename the necessary 8 files according to your language/script. For Bangla we used lang="ban" and for Devanagari we used lang="dev". So, the name of the necessary 8 files will be prefixed by lang+'.' (Example: ban.unicharset, dev.unicharset). These 8 files must be copied into the tessdata subdirectory if these are generated any other place.

Testing tesseract

Required files
To test Bangla or Devanagari scripts (lang = ban/dev), we have two files in ocropus-0.2/ocroscript subdirectory. Those files are:

ocropus-0.2/ocroscript/rec-ltess.lua
ocropus-0.2/ocroscript/rec-tess.lua

Among these two lua files rec-ltess.lua is used mainly to observe the performance of ocropus layout analysis and rec-tess.lua is used to observe the performance of character recognition.

Usage of the lua files for testing

rec-ltess.lua

```
if #arg <>
arg = { "../../data/pages/alice_1.png" }
end
pages = Pages()
pages:parseSpec(arg[1])
segmenter = make_SegmentPageByRAST()
page_image = bytearray()
page_segmentation = intarray()
line_image = bytearray()
bboxes = rectanglearray()
costs = floatarray()
tesseract_recognizer = make_TesseractRecognizeLine()
tesseract.init("ban")
while pages:nextPage() do
```

```
pages:getBinary(page_image)
segmenter:segment(page_segmentation,page_image)
regions = RegionExtractor()
regions:setPageLines(page_segmentation)
for i = 1,regions:length()-1 do
regions:extract(line_image,page_image,i,1)
fst = make_StandardFst()
tesseract_recognizer:recognizeLine(fst,line_image)
result = nustring()
fst:bestpath(result)
print(result:utf8())
end
end
```

rec-tess.lua

```
require 'lib.util'
require 'lib.headings'
require 'lib.paragraphs'
if not tesseract then
print "Compiled without Tesseract support, can't continue."
os.exit(1)
end
opt,arg = getopt(arg)
if #arg == 0 then
print "Usage: ocroscript rec-tess [--tesslanguage=...] input.png ... >output.hocr"
os.exit(1)
end

set_version_string(hardcoded_version_string())

segmenter = make_SegmentPageByRAST()
page_image = bytearray()
page_segmentation = intarray()
```

```
function convert_RecognizedPage_to_PageNode(p)
page = PageNode()
page.width = p:width()
page.height = p:height()
page.description = p:description()
for i = 0, p:linesCount() - 1 do
local bbox = p:bbox(i)
local text = nustring()
p:text(text, i)
page:append(LineNode(bbox, text))
end
return page
end


document = DocumentNode()
for i = 1, #arg do
pages = Pages()
pages:parseSpec(arg[i])
while pages:nextPage() do
pages:getBinary(page_image)
segmenter:segment(page_segmentation,page_image)
local p = RecognizedPage()
tesseract_recognize_blockwise(p, page_image, page_segmentation)
p = convert_RecognizedPage_to_PageNode(p)
p.description = pages:getFileName()

local regions = RegionExtractor()
regions:setPageLines(page_segmentation)
p.headings = detect_headings(regions, page_image)
p.paragraphs = detect_paragraphs(regions, page_image)
document:append(p)
```

end

end

document:hocr_output()

Note: At present in ocropus-0.2 this file is suffering from the problem of representing the unicode output for Bangla as well as Devanagari. The reason is that the language specified in this file is overwritten in the file tesseract.cc. So, we edit the file tesseract.cc as follows:

namespace ocropus {
param_string tesslanguage("tesslanguage", "**ban**", "Specify the language for Tesseract");
}

Now the problem of unicode representation is solved.

Result Analysis and discussion

We perform the result testing in two steps. In the first step we observed performance of tesseract to recognize the isolated characters. The accuracy was reported as 98% in recognition of the basic characters with different font. Depending on the result of this step we decide the amount of training data units. In the second step we performed the testing with page image where we considered images with same font as well as different. The maximum accuracy was reported up to 92% in the testing. Table 3 represents the results of out testing.

Type of image	**Font**	**Accuracy**
Isolated character	same	99.5%
Isolated character	different	97%
Word image	same	93.5%
Word image	different	88%

Table 3: Experimental result

CRBLP segmenter plugin

Four step character segmentation algorithm.
Projection profile based.
Elimination of the joining errors
Shadow character segmentation
Touching character segmentation
Elimination of splitting errors
Reordering the modifiers

How to add a method in OCROPUS-0.2

i added a method in in file grouping.cc
method name is : crblpRecognizeLine(....)

i did it as the fellowing steps:

1. added my method body in the file grouping.cc in
struct NewGroupingLineOCR : IRecognizeLine {..};

2. added a fake body in file tesseract.cc in
struct TesseractRecognizeLine : IRecognizeLine {..};

as this: void crblpRecognizeLine(...){}

3. added a fake body in file glinerec.cc in
struct GLineRec : IRecognizeLine {..};

as this: void crblpRecognizeLine(...){}

4. added a prototype in file ocrinterfaces.h in
struct IRecognizeLine : IComponent {..};

as this: virtual void crblpRecognizeLine(...)=0;

5. added a prototype in file ocr.pkg in
struct IRecognizeLine : IComponent {..};

as this: virtual void crblpRecognizeLine(...)=0;

now compile
you can now call it from cpp or lua

Conclusion

In this thesis I added Bengali script support in OCRopus and tested the performance of its two recognition engine one is tesseract and other is bpnet and added CRBLP segmentation plug-in in OCRopus. At this point of development the result is not so satisfactory in all test cases but better then any existing system for Bengali . So, farther research and development on this system can give very satisfactory result for Bengali script recognition.

References

http://code.google.com/p/ocropus/
http://code.google.com/p/tesseract-ocr/
http://code.google.com/p/ocropus-bengali/
http://crblpocr.blogspot.com/