

	Shape		Shape	
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

In the Bangla script, a consonant followed by another consonant often takes a compound shape, which we call a compound character. Examples of a few compound characters are shown in Table 2. There are around 270 compound characters present in the Bangla script [5], most of which are formed by consonant-consonant combinations; compounding of three consonants is also possible. Among the listed compound characters, there are a few characters that do not change their respective shapes; instead these just attach a consonant modifier similar to the case of the vowel modifiers. There are 3 consonant modifiers that completely change their shapes. These compound characters can also be followed by dependent vowels.

Table 2: Example of compound characters

Consonant Combination	Compound character
ক্ ক	ক্ক
ক্ ত	ক্কত
ক্ ন	ক্কন
ক্ ম	ক্কম
ক্ ষ	ক্কষ
ঙ্ ক	ক্ক
ক্ ল	ক্কল
ল্ ক	ক্ক

After an analysis of Tesseract, we found, that to integrate the Bangla script recognition support in Tesseract, we need a complete set of training data. This training data is dependent on the output of the script segmenter which will be included during test data recognition. It also depends on the nature of the segmenter that is included in the Tesseract engine; the engine has its own segmenter to detect lines, words

and characters. Since Tesseract is a complete OCR package, once it is trained with the training data, we do not need to be concerned about feature extraction and recognition in order to recognize the characters in the Bangla script.

To the best of our knowledge this is the first reported attempt to integrate Bangla script recognition in Tesseract. We briefly present our methodology in section 2, followed by results with discussion in section 3, and then conclude.

2. Methodology

The entire task is divided into two parts:

1. Training data generation
2. Test data processing

2.1. Training data generation

A basic guideline to prepare training data is nicely written in [2], which we followed to prepare the training data. However, there are several problems that we found during training data preparation. The problems and our solutions are described below.

2.1.1. Prepare training data Image.

As is probably evident by now, Bangla, along with other members of the Brahmi family of scripts, is a fairly complex script. For English, Tesseract was trained with just 94 characters/units [3]. We listed 340 (50 basic, 10 vowel modifiers and 270 compound characters) characters, and considered these as the basic units for training. We performed our experiment with the training data to estimate of the necessary amount of training data. For this experimental purpose, we considered two approaches as follows:

1. Train dependent modifiers separately than the basic units.
2. Train dependent modifiers combined with the basic units.

In a word image the modifiers (vowel & consonant) are placed around the core characters and often connected with the basic characters following few certain characteristics. In most of the cases the image of basic character and modifier has a certain amount of overlap between them. As a result it is impossible to make a horizontal and vertical boundary line between them. This scenario is completely different than English character where there are no modifiers as well as the amount of overlapping and touching problem is very less. We observed the performance of the trained units generated from the first approach. We noted that Tesseract can successfully recognize those units which the character segmenter is capable to isolate properly

the characters with the value equivalent to uppercase letter. The output of this process is a *.unicarset file. An example of character set file is shown in figure-6.

Figure 6: Example of a character set file

```

@ 5 NULL
@ 5 NULL
o 8 NULL
s 8 NULL
z 8 NULL
u 8 NULL

```

2.1.6. 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 we used wordlist2dawg program. We collected a unique word list of 180K words and frequent word list of 30K words. Using the word list and the frequent word list we generate the dictionary files freq-dawg and word-dawg. In the third dictionary file called user-words we put all the characters.

2.1.7. Prepare DangAmbigs file.

To prepare this file we identified the possible intrinsic ambiguity between characters or sets of characters during recognition. This file can be generated from the erroneously generated output. Information about a number of possible ambiguities is found from possible OCR errors listed in the literature [6]. Examples of few ambiguities are shown in figure-7.

Figure 7: Intrinsic ambiguity between characters

1	কা	2	ক গ
2	ক গ	1	কা
2	তা	1	অ

After finished the generation of all the training data files (8 files) we renamed each files according to our language/script which is “ban” (Example: ban.unicarset). These 8 files must be copied into the tessdata subdirectory.

2.2. Test data processing

The main goal of this step is to prepare the test image in a form suitable for Tesseract to process and recognize. Tesseract is a raw OCR engine, so it does not include a preprocessor, and so we need to perform the preprocessing task at this stage. We divide the test data processing task into four main parts like any other OCR as follows:

1. Binarization
2. Noise elimination
3. Skew detection and correction
4. Character segmentation

To perform tasks 1-3, we followed the approaches described in [6]. To perform segmentation, we followed the approach that is specific to the requirement of the Tesseract engine. We performed an experiment to recognize isolated character image using our training data to identify the requirements of the Tesseract recognizer. From the observed output, it is clear that the recognition rate will be very high if it can successfully extract the bounding box of each character/unit from the text image. We also note that Tesseract can successfully recognize units which are broken into two parts. These observations make the requirement of the segmenter very clear which is that the segmenter needs to separate each unit from its left or right units using a horizontally separable line. There are several segmentation algorithms available in the literatures [6-8] to perform segmentation. We followed the algorithms mentioned in [7,8], but without segmenting the upper and lower modifiers. The output of the segmenter is shown in Fig. 8. The segmenter output is then passed on to the Tesseract recognition engine and the output test is saved in a text file.

Figure 8: Output of the segmenter. Upper one is input image and the lower one is the segmented image



3. Result Analysis and discussion

We tested our results in two steps. In the first step, we observed the performance of Tesseract in recognizing isolated characters. The accuracy in recognizing the basic characters with different font was 98%. The result of this step also provided us with the amount of training data that we needed to create. In the second step, we performed the testing with page images where we considered images with same and different fonts, resulting in a maximum accuracy of 92%. Table 3 summarizes our testing results.

Table 3: Experimental result

Type of image	Font	Accuracy
Isolated character	Same	99.5%
Isolated character	Different	97%
Word image	Same	93.5%

Word image	Different	88%
------------	-----------	-----

The generated output of the recognizer on test image (figure 8) is “শহরেকন্দিক জীবন গেড় উঠলেও বাংলাদেশের গ্রামের জীবনই প্রকৃত বৈশিষ্ট্যের অধিকারী”. We can see from the output that the erroneous outputs “জী” and “জী” are occurring instead of “জী” which is very close. To identify the errors in recognition, we performed several tests with the dictionary data and identified that these are not contributing to the recognition accuracy, at least in the case of the Bangla script. We will investigate extending the language modeling capability for Brahmi scripts in the future.

4. Conclusion

In this paper, we present a complete procedure to Bangla printed text open source Tesseract OCR engine. We first conducted a series of tests to decide how much training data we needed, and to understand the requirements of the Tesseract's built-in segmenter. We then trained Tesseract's recognizer with the Bangla training data, and then observed the results using scripts of different sizes and fonts. The best results were when using the same font as the training data, but the result is quite promising when using a different font. The methodology presented here would be applicable. Improvements in the segmenter, and more data in the training set would improve the overall quality for all inputs which is something we would be working on next. The methodology outlined here would also be just as applicable to other members of the Brahmi family of scripts, such as Devanagari.

6. Acknowledgements.

This work has been supported in part by the PAN Localization Project (www.PANL10n.net) grant from the International Development Research Center, Ottawa, Canada.

5. References

- [1]. <http://google-code-updates.blogspot.com/2006/08/announcing-Tesseract-ocr.html> last accessed 12 August, 2008.
- [2]. <http://code.google.com/p/Tesseract-ocr/> last accessed 12 August 2008.
- [3]. Ray Smith, "An Overview of the Tesseract OCR Engine", Proc. of ICDAR 2007, Curitiba, Paraná, Brazil, 2007.
Available:
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4376991
- [4]. S. M. Murtoza Habib, N. A. Noor and Mumit Khan, "Skew correction of Bangla script using Radon Transform", Proc. of ICCIT'06, Dhaka, Bangladesh, December, 2006.
Available:
<http://www.panl10n.net/english/final%20reports/pdf%20files/Bangladesh/BAN02.pdf>
- [5] Muhammod Enamul Hoque, Sibprosonno Lahiri, Shorochish Sarkar, "Bangla Academy Byabharik Bangla Abhidhan", Bangla Academi Press, Dhaka, Bangladesh, September 2003.
- [6] Md. Abul Hasnat, S M Murtoza Habib and Mumit Khan. "A high performance domain specific OCR for Bangla script", Proc. of CISSE'07, 2007.
Available:
www.springerlink.com/index/175400676825p373.pdf
- [7] U. Pal, B. B. Chaudhuri, "OCR in Bangla: an Indo-Bangladeshi Language", Proc. of ICPR, pp. 269-274, Jerusalem, Israel, 1994.
Available:
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=576917
- [8] B. B. Chaudhuri, U. Pal, "An OCR System to Read Two Indian Language Scripts: Bangla and Devnagari(Hindi)", Proc. of 4th ICDAR, Page(s): 1011 -1015 vol.2, Ulm, Germany, 1997.
Available:
<http://ieeexplore.ieee.org/Xplore/login.jsp?url=/iel3/4891/13496/00620662.pdf?arnumber=620662>