Integrating Bangla Script Recognition Support in Tesseract OCR

Md. Abul Hasnat, Muttakinur Rahman Chowdhury and Mumit Khan
Center for Research on Bangla Language Processing, Department of Computer Science and Engineering, BRAC University, Dhaka, Bangladesh
mhasnat@gmail.com, sourogts@gmail.com, mumit@bracu.ac.bd

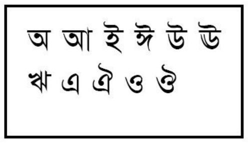
Abstract

Tesseract is considered one of the most accurate free software OCR engines currently available. It was originally developed by Hewlett-Packard from 1985 until 1995, and is currently maintained by Google. At present, Tesseract is capable of only recognizing English, French, Italian, German, Spanish and Dutch. However, it is possible to make Tesseract recognize other scripts if the engine is trained with the requisite data. In this paper, we present a complete methodology to integrate Bangla script recognition support in Tesseract.

1. Introduction

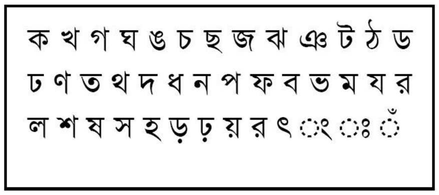
The Tesseract OCR engine was one of the top 3 engines in the 1995 UNLV Accuracy Test. Between 1995 and 2006 however, there was very little activity in Tesseract, until it was open-sourced by HP and UNLV in 2005; it was again re-released to the open-source community in August of 2006 by Google [1]. Today, it is considered one of the most accurate open source OCR engines available. A complete overview of Tesseract OCR engine can be found in [3]. While Tesseract was originally developed for English, it has since been extended to recognize French, Italian, German, Spanish and Dutch. It has the ability to train in other languages and scripts as well [2]. The algorithms used in the different stages of the engine are specific to English alphabet, which makes it easier to support other scripts that are structurally similar by simply training the character set from the new scripts. However, for Brahmi family of scripts like Bangla and Devanagari, one must also consider the issue of line and character segmentation, as well as how to create the training data. In-depth knowledge of the Bangla and Devanagari scripts is necessary to integrate the recognition support for these scripts in Tesseract.

In the Bangla script, there are 11 independent vowel and 39 consonant characters. These are called the basic characters, as shown in Fig. 1. From Fig. 1, it can be seen 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 the 50 basic characters, 32 characters have this matra (head line). Out of these 11 independent vowels, 10 vowels have dependent forms (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 in its independent form [4]. Generally, a vowel following by a consonant takes a dependent form, and the dependent form of the vowel is placed at the left or the right or both, or at the bottom of the consonant.



অ আ ই ঈ উ ঊ
ঋ এ ঐ ও ঔ

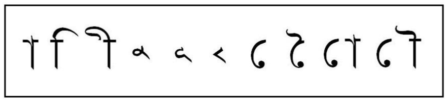
(a) Vowels



ক খ গ ঘ ঙ চ ছ জ ঝ ঞ ট ঠ ড
ঢ ণ ত থ দ ধ ন প ফ ব ভ ম য র
ল শ ষ স হ ড় ঢ় য় র ৎ ং ঃ ঁ

(b) Consonants

Figure 1: Basic characters of the Bangla script



া ি ী ু ূ ৃ ে ৈ ো ৌ

Figure 2: Dependent vowels

Table 1: Example of independent and dependant vowels

	Indep. shape	Example	Dep. shape	Example
1.	আ	আমরা	◌া	কাল
2.	ই	ইন্দুর	◌ি	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
ক ◌্ক	ক্ক
ক ◌্ত	ক্ত
ক ◌্ন	ক্ন
ক ◌্ম	ক্ম
ক ◌্ষ	ক্ষ
ঙ ◌্ক	ঙ্ক
ক ◌্ল	ক্ল
ল ◌্ক	ল্ক

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 farily 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 from the surrounding units. If the segmenter is successful in separating the modifier (example: া, ে, ৈ ্য) and basic characters using a vertical column then Tesseract is successful in recognition. However in case of other modifiers (Example: ি, ী, ু, ূ, ৃ, ৌ, ঁ and ্র) it is not possible for the segmenter to separate them using any vertical column. In such cases Tesseract failed to recognize them because it is unable to identify the bounding box for modifier and basic unit. This observation motivates us towards approach-2 where it is necessary to train all possible combinations of the modifiers (Example: ি, ী, ু, ূ, ৃ, ৌ, ঁ and ্র) and basic characters. An example of such a scenario of few training units with modifier is shown in figure 3.

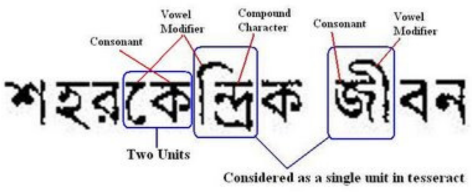


Figure 3: Example of few training units where basic character is overlapped by modifiers

So, in the final training data set will consider the following combinations:

- All vowels, consonants and numerals
- Consonants + vowel modifiers
- Consonants + consonant modifiers
- Combined consonants (compound character)
- Compound characters + vowel modifiers
- Compound characters + consonant modifiers

Following approach-2 the total amount of training data units will be around 3200. Next we typed all the combinations and take print out of the documents (13 pages for a single font). Next we scan the pages and manually preprocessed the pages which include skew correction. We choose the most popular fonts that have been widely used for a long time to print Bangla documents, as maximum documents that we are targeting to recognize is written in those fonts. Most of the popular fonts are non unicode which is a problem because it cannot be used for transcription. So, we used unicode font (Solaimanlipi) to prepare the transcription. An example of a part of the training image is shown in figure-4.

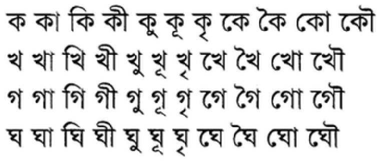
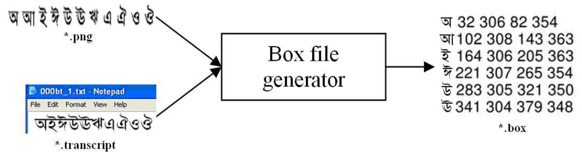

ক কা কি কী কু কূ কৃ কে কৈ কো কৌ
খ খা খি খী খু খূ খৃ খে খৈ খো খৌ
গ গা গি গী গু গূ গৃ গে গৈ গো গৌ
ঘ ঘা ঘি ঘী ঘু ঘূ ঘৃ ঘে ঘৈ ঘো ঘৌ

Figure 4: Part of the training data generated using SutonnyMJ font

2.1.2. Prepare box file. A box file is a text file that contains necessary information about the bounding box of each character/unit in a training image. Each training image must have a box file where the number of box information must match with number of training character/unit in the image. Tesseract has its own tool to create box information from an image which create box file considering the nature of English character. As a result it fails to correctly generate box information from Bangla character image in several cases where a modifier is slightly detached from the core character. Another point is that we have to manually correct the transcription in the created box file by Tesseract which is a very monotonous task. To handle these issues we create our own box file generator. The input of our box file generator is a training image and a transcription file of the image and the output will the box file (Figure. 5).



অইঈউঊঋএঐওঔ

```
অ 32 306 82 354
আ 102 308 143 363
ই 164 306 205 363
ঈ 221 307 265 354
উ 283 305 321 350
ঊ 341 304 379 348
```

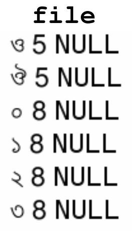
Figure 5: Structure of the box file generator

2.1.3. Prepare feature files. We performed this task by invoking the Tesseract engine in training mode. The output is a *.tr file which contains the features of each character of the training page.

2.1.4. Prepare clustered files.

We perform the clustering using two different programs called mftraining and cntraining. This clustering is performed to create the prototypes using the character shape features. The output of this step is *.inttemp and *.pffmtable file.

2.1.5. Prepare character set file. To create the character set file we used the program called unicharset_extractor. This program takes the *.box files as input. Next we manually set the properties (isalpha, isdigit, isupper and islower) of each character in the file. As Bangla characters do not have uppercase and lowercase variation so we set all the characters with the value equivalent to uppercase letter. The output of this process is a *.unicharset file. An example of character set file is shown in figure-6.

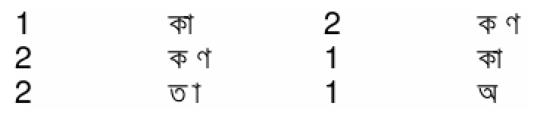


```
file
ও 5 NULL
ঔ 5 NULL
০ 8 NULL
১ 8 NULL
২ 8 NULL
৩ 8 NULL
```

Figure 6: Example of a character set

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.



1	কা	2	ক ণ
2	ক ণ	1	কা
2	তা	1	অ

Figure 7: Intrinsic ambiguity between characters

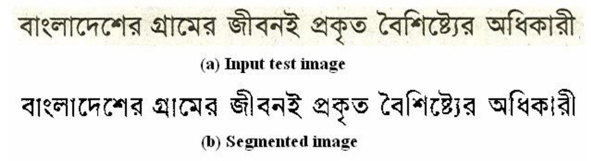
After finished the generation of all the training data files (8 files) we renamed each files according to our language/script which is “ban” (Example: ban.unicharset). These 8 files must be copied into the tessdata subdirectory.

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.



বাংলাদেশের গ্রামের জীবনই প্রকৃত বৈশিষ্ট্যের অধিকারী

(a) Input test image

বাংলাদেশের গ্রামের জীবনই প্রকৃত বৈশিষ্ট্যের অধিকারী

(b) Segmented image

Figure 8: Output of the segmenter (a) Input image (b) 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 "শহরেকন্দ্রিক জীঁবন গেড় উঠলেও বাংলাদেশের গ্রামের র্জীবনই প্রকৃত বৈশিষ্ট্যের অধিকারী". We can see from the output that the erroneous outputs "র্জী" and "জীঁ" 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 Tessearct'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 Devanagri.

5. 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.

6. 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 in proc. *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", in proc. *ICCIT'06,* Dhaka, Bangladesh, December, 2006.
Available:
http://www.panl10n.net/english/final%20reports/pdf%20files/Bangladesh/BAN02.pdf

[5] M. E. Hoque, S. Lahiri, S. Sarkar, "Bangla Academy Byabaharik 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/l75400676825p373.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