How to train Bangla and Devanagari script for tesseract engine

Introduction

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:

- `tessdata/lang.freq-dawg`
- `tessdata/lang.word-dawg`
- `tessdata/lang.user-words`
- `tessdata/lang.inttemp`
- `tessdata/lang.normproto`
- `tessdata/lang.pffmtable`
- `tessdata/lang.unicharset`
- `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).

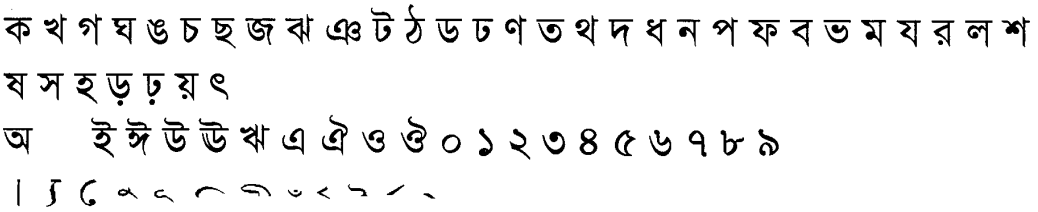

Figure-1: Training data units for Bangla script

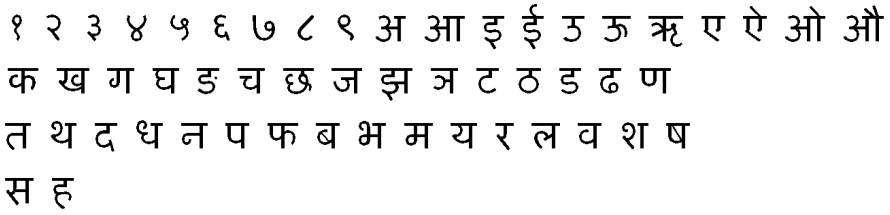

Figure-2: Training data units for Devanagari script

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.

```
ক 32 306 82 354
খ 102 308 143 363
গ 164 306 205 363
ঘ 221 307 265 354
ঙ 283 305 321 350
চ 341 304 379 348
ছ 396 294 440 349
জ 457 303 510 351
ঝ 527 303 577 357
ঞ 598 301 660 349
ট 677 304 719 365
ঠ 736 300 773 371
১ 37 338 54 377
২ 88 338 111 377
৩ 148 337 170 377
৪ 206 337 237 376
৫ 270 338 300 376
৬ 334 338 359 377
৭ 391 338 426 376
৮ 457 338 483 376
৯ 516 338 542 377
অ 572 337 618 377
আ 644 337 705 377
ই 729 327 763 376
```

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.

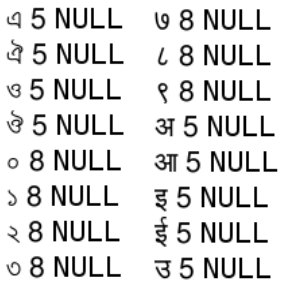

Figure-4: unicharset file for Bangla and Devanagari script

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.

1	কা	2	ক ণ
2	ক ণ	1	কা
2	তা	1	অ

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.