Adaptation of Manipuri OCR FROM Bengali TESSERACT – OCR

Summer Internship 2019
IIT Guwahati
Dept. Computer Science and Engineering

Sanajaoba Thongram
NIT Manipur, CSE
Ningthoujam Justwant Singh
NIT Manipur, CSE

Under the Guidance of:

Sanasam Ranbir Singh
Associate Professor

Jennil Thiyam
PhD Research Scholar

Department of Computer Science and Engineering
Indian Institute of Technology Guwahati

ACKNOWLEGDEMENT

It is with a sense of gratitude, we acknowledge the efforts of entire hosts of well-wishers who have in some way or other contributed in their own special ways to the success and completion of this summer internship project.

First of all, we express our sage sense of gratitude and indebtedness to our internship supervisor and project guide **Assoc. Prof. Sanasam Ranbir Singh** of **IIT-Guwahati**, **Department of Computer Science and Engineering**, for his immense support and faith upon us.

We would also like to thank our project mentor **Mr. Jennil Thiyam, Ph.D. Scholar, IIT Guwahati**, **Centre For Linguistic Science and Technology**, for his kind advice, suggestions and constant help in a lot of ways during the course of this summer internship, without which we wouldn't be able to complete this project.

Further, we would like to express our gratitude to all the scholars and staffs of **OSINT** lab and **CLST**, **IIT Guwahati** for their valuable suggestions, corporation, motivation, encouragement, and support in many ways which help in the completion of this summer internship project.

Contents

1. Introduction

The Meitei or Manipuri language is one of the scheduled languages (included in the 8th schedule by the 71st amendment of the constitution in 1992). It is a Sino-Tibetan language and the predominant language and lingua franca in the southeastern Himalayan state of Manipur, in northeastern India. It is the official language in government offices. Meitei is also spoken in the Northeast Indian states of Assam and Tripura, and in Bangladesh and Burma (now Myanmar). It is currently classified as a vulnerable language by UNESCO.

Meitei has its own script, which was used until the 18th century. Its earliest use is not known. Pamheiba, the ruler of the Manipur Kingdom who introduced Hinduism, banned the use of the Meitei script and adopted the Bengali script. Between 1709 and the middle of the 20th century, the Meitei language was written using the Bengali script. So, most of the literature and historical records were written in Bengali script. Now in schools and colleges, the Bengali script is gradually being replaced by the Meitei script. And a major transliteration is in need to reproduce those documents in Meitei Mayek script.

There are few OCR model for Bengali script, unfortunately, none (to be specific a Tesseract-OCR model) works for Manipuri (Bengali script) with acceptable accuracy. An efficient OCR-model could be helpful in speeding up the transliteration and archival process.

And to develop an efficient OCR system for Manipuri (Bengali script) is the objective of our project. Due to similarity between the script used for Bengali and Manipuri, the OCR model (Manipuri) will be adapted on the existing Bengali – Tesseract – OCR model using fine tuning.

2. Bengali Script and Manipuri

Vowels

In Bengali Alphabet, there is a total of 11 vowel symbols, whereas in the Bengali – Manipuri alphabet there are 14 vowel symbols in total.

Manipuri Vowels:

অ	আ	ই	ঈ
উ	ঊ	ঋ	ঌ
এ	ঐ	ও	ঔ
	অং	অঃ	

Bengali Vowels:

অ	আ	ই	ঈ
উ	ঊ	ঋ	
এ	ঐ	ও	ঔ

Consonants

In Bengali Alphabet, there is a total of 35 consonant symbols, whereas in the Bengali Manipuri alphabet there are 41 consonant symbols in total.

Bengali Vowel symbols used in written Manipuri are as follow:

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	ৱ	শ
ঐ	স	হ	ক্ষ	ড়
ঢ়	য়	ং	ঃ	ঁ
		ৎ		

Bengali script Consonants are as follows:

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

Numerals

০	১	২	৩	৪
0	1	2	3	4
৫	৬	৭	৮	৯
5	6	7	8	9

Consonant along with the **diacritic** form of the vowels অ, আ, ই, ঈ, উ, ঊ, ঋ, এ, ঐ, ও and ঔ:

	ô	a	i	ī	u	ū	ṛ	e	oi	o	ou
kô	ক	কা	কি	কী	কু	কূ	কৃ	কে	কৈ	কো	কৌ
khô	খ	খা	খি	খী	খু	খূ	খৃ	খে	খৈ	খো	খৌ
gô	গ	গা	গি	গী	গু	গূ	গৃ	গে	গৈ	গো	গৌ
ghô	ঘ	ঘা	ঘি	ঘী	ঘু	ঘূ	ঘৃ	ঘে	ঘৈ	ঘো	ঘৌ
ṅô	ঙ	ঙা	ঙি	ঙী	ঙু	ঙূ	ঙৃ	ঙে	ঙৈ	ঙো	ঙৌ

[...]

hô	হ	হা	হি	হী	হু	হূ	হৃ	হে	হৈ	হো	হৌ
yô	য়	য়া	য়ি	য়ী	য়ু	য়ূ	য়ৃ	য়ে	য়ৈ	য়ো	য়ৌ
rô	ড়	ড়া	ড়ি	ড়ী	ড়ু	ড়ূ	ড়ৃ	ড়ে	ড়ৈ	ড়ো	ড়ৌ
rhô	ঢ়	ঢ়া	ঢ়ি	ঢ়ী	ঢ়ু	ঢ়ূ	ঢ়ৃ	ঢ়ে	ঢ়ৈ	ঢ়ো	ঢ়ৌ

Conjuncts

Bengali is full of conjunct consonants. Here are some of them:

ক্ক kkô	ক্ট kṭô	ক্ত ktô	ক্ব kbô	ক্ম kmô	ক্র krô	ক্ল klô	ক্ষ kṣô	ক্ষ্ম kṣmô
ক্স ksô	গ্ধ gdhô	গ্ন gnô	গ্ব gbô	গ্ম gmô	গ্ল glô	গ্ল ghnô	ঙ্ক ṅkô	ঙ্ক্ষ ṅkṣô
ঙ্থ ṅthô	ঙ্গ ṅgô	ঙ্ঘ ṅghô	ঙ্ম ṅmô	চ্ছ cchô	চ্ছ্ব cchbô	ঞ্চ cñô	জ্জ jjô	জ্জ্ব jjbô
জ্ঝ jjhô	জ্ঞ jñô	জ্ব jbô	ঞ্চ ñcô	ঞ্ছ ñchô	ঞ্জ ñjhô	ট্ট ṭṭô	ট্ব ṭbô	ণ্ট ṇṭô
ণ্ঠ ṇṭhô	ণ্ড ṇḍô	ণ্ন ṇṇô	ণ্ম ṇmô	ত্ত ttô	ত্ত্ব ttbô	ত্থ tthô	ত্ন tnô	ত্ব tbô
ত্ম tmô	ত্র trô	দ্দ ddô	দ্ধ ddhô	দ্ব dbô	দ্র dbhrô	ন্ট nṭô	ন্ড nḍô	ন্ত ntô
ন্ত্ব ntbô	ন্ত্র ntrô	ন্দ ndô	ন্ধ ndhô	ন্ন nnô	ন্ব nbô	ন্স nsô	প্ট pṭô	প্ত ptô
প্ন pnô	প্প ppô	প্ল plô	প্স psô	ফ্ল phlô	ভ্র bhrô	ভ্ল bhlô	ম্ন mnô	ম্ফ mphô
ম্ব mbô	ম্ল mlô	ল্ট lṭô	ল্ড lḍô	ল্ব lbô	ল্ল llô	শ্চ shchô	ষ্ক ṣkô	ষ্ট ṣṭô
ষ্ণ ṣṇô	স্ক্র skrô	স্ত stô	স্ত্র strô	স্ব sbô	হ্ন hnô	হ্ম hmô	হ্ব hbô	হ্ল hlô

Some of the Manipuri Conjunts are as follows:

ক্য	ক্র	ক্ল	ক্ব	হ্ন	ক্ম	র্ক	ক্ক	ক্ট্র
খ্য	খ্র	প্ল	খ্ব	ষ্ণ	জ্ঞ	র্খ	ক্ত	জ্ঞ্র
জ্য	দ্র	ম্ল	গ্ব	ক্ল	ত্ম	র্গ	ক্ষ	জ্ঞ্র
ন্য	ত্র	ল্ল	দ্ব	ত্ন	ন্ম	র্জ	ক্স	ন্ত্র
প্য	থ্র	হ্ল	শ্ব	ন্ন	ম্ম	র্ত	ঙ্ক	থ্র
ব্য	প্র	চ্চ	স্ব	প্ন	ল্ম	র্থ	ঙ্খ	প্ত্র
শ্য	ভ্র	ঞ্চ	ট্ব	ম্ন	স্ম	র্প	ঙ্গ	ম্র
হ্য	হ্র	জ্ঞ	ণ্ঠ	হ্ন	শ্ম	র্ফ	ঙ্ঘ	ন্ধ্য
ঙ্য	ঙ্র	ঞ্জ	ণ্ড	ন্স	ষ্ম	র্ভ	ঙ্ক	ক্ষ্ম
ত্য	জ্র	চ্ছ	ন্দ	শ্ন	ক্ষ্ম	র্ম্ম	ত্ত	ন্দ্র
ব্র	গ্র	ঞ্ছ	ণ্ড	ল্ট	ন্ত	র্ত্ত্য	থ্ব	স্প্র
স্ফ	দ্র	ষ্ট	প্প	শ্ন	হ্ন	র্দ্ধ্য	দ্দ	ন্ধ
স্ব	ফ্র	হ্ন	প্স	স্প				

3. Tesseract – OCR

Tesseract was originally developed at Hewlett-Packard Laboratories Bristol and at Hewlett-Packard Co, Greeley Colorado between 1985 and 1994, with some more changes made in 1996 to port to Windows, and some C++izing in 1998. In 2005 Tesseract was open sourced by HP and the University of Nevada, Las Vegas (UNLV)[1]. Since 2006 it is developed by Google.

The latest (LSTM based) stable version is **4.0.0**, released on October 29, 2018.

3.1 Why Tesseract – OCR?

Tesseract is an OCR engine with support for **unicode (UTF-8)** and the ability to recognize **more than 100 languages** out of the box. It can be trained to recognize other languages.

Tesseract supports **various output formats**: plain text, hOCR (HTML), PDF, invisible-text-only PDF, TSV. The master branch also has experimental support for ALTO (XML) output. And this OCR engine has a higher accuracy than its counterparts.

3.2 Installing Tesseract [linux]

To install **Tesseract 4.x** you can simply run the following command on your **Ubuntu 18.xx bionic**:

```
sudo apt install tesseract-ocr
```

To install the **Developer Tools** which can be used for training, run the following command:

```
sudo apt install libtesseract-dev
```

3.3 Fine Tune

Fine tuning is one of the options provided by Tesseract engine to start with an existing trained language, train on our own specific additional data. This may work for problems that are close to the existing training data, but different in some subtle way, like a particularly unusual font. May work with even a small amount of training data.

There are two main types of Fine Tuning in Tesseract 4.xx.

1. Fine tuning for specific fonts.
2. Fine tuning for additional symbols.

4. Requirements

4.1 Tools

1. Tesseract 4.1.0-rc4

   https://github.com/tesseract-ocr/tesseract/releases/tag/4.1.0-rc4

2. GPL Ghostscript 9.26 (2018-11-20)
3. OCR-evaluation-Tools by UNLV [1].

   https://github.com/Sanaj2060/ocr-evaluation-tools

4.2 Dependencies

1. A compiler for C and C++: GCC or Clan
2. GNU Autotools: autoconf, automake, libtool
3. pkg-config
4. Leptonica
5. libpng, libjpeg, libtif

4.3 Data

1. tesseract-ocr/tessdata_best/ben.traineddata

   https://github.com/tesseract-ocr/tessdata_best/blob/master/ben.traineddata

2. tesseract-ocr/tessdata_best/script/Bengali.traineddata

   https://github.com/tesseract-ocr/tessdata_best/blob/master/script/Bengali.traineddata

3. langdata_master

   https://github.com/tesseract-ocr/langdata.git

4. langdata_lstm_master

   https://github.com/tesseract-ocr/langdata_lstm.git

5. Manipuri text from Naharol gi Thoudang, a Manipuri newspaper.

   https://github.com/Sanaj2060/tesseract-ocr-Manipuri-Bengali/blob/master/data/manipuri.txt

6. Manipuri Conjuncts from **Maapi Lyrik**, a Manipuri Primer.

   https://github.com/Sanaj2060/tesseract-ocr-Manipuri-Bengali/blob/master/data/conjuncts.txt

5. Creating Starter Traineddata

A **starter traineddata** file is given during training, and has to be setup in advance. It can contain:

- Config file providing control parameters.
- **Unicharset** defining the character set.
- **Unicharcompress,** aka the recoder, which maps the unicharset further to the codes actually used by the neural network recognizer.
- Punctuation pattern dawg, with patterns of punctuation allowed around words.
- Word dawg. The system word-list language model.
- Number dawg, with patterns of numbers that are allowed.

Bold elements **must** be provided. Others are optional, but if any of the dawgs are provided, the punctuation dawg must also be provided. A new tool: `combine_lang_model` is provided to make a **starter traineddata** from a *unicharset* and optional *wordlists*.

5.1 Modification of ben.training_txt

We select few words from the collection of Manipuri text **[4.3.5]** to make sure that the text to be added has every unique symbols of Manipuri (Bengali Script) and all the conjuncts from the primer **[4.3.6],** then append it to:

1. langdata_master/ben/ben.training_txt **[4.3.3]**
2. langdata_lstm_master/ben/ben.training_txt **[4.3.4]**

The config and other optional files are same with the one used to train the **ben.traineddata**.

5.2 Running tesstrain.sh

Training data is created using *tesstrain.sh* as follows:

```
src/training/tesstrain.sh --fonts_dir /usr/share/fonts --lang ben
--linedata_only \
--noextract_font_properties --langdata_dir ../langdata \
--fontlist “Bangla Medium” “Lohit Bengali” \
--tessdata_dir ./tessdata --output_dir ~/tesstutorial/ben_man
```

We run *tesstrain.sh* on the modified ben.training_txt **[5.1.1]** and **[5.1.2]**, seperately using two fonts, Bangla Medium and Lohit Bengali each.

And each output has 6 files and are as follows:

1. ben.Bangla_Medium.exp0.lstmf
2. ben.Lohit_Bengali.exp0.lstmf
3. ben.training_files.txt
4. ben.charset_size=xyz.txt
5. ben.traineddata
6. ben.unicharset

* *tesstrain.sh* also generate tiff and box files from the training_txt.

6. Fine Tuning on the new training data

In tesseract-ocr 4.xx we can fine tune a existing model using:

```
[i]   training/combine_tessdata -e tessdata/lang.traineddata \
        ~/lang.lstm

[ii]  training/lstmtraining --model_output /path/to/output [--max_image_MB
      6000]\
        --continue_from /path/to/existing/model \
        --traineddata /path/to/traineddata/with/new/unicharset \
        --old_traineddata /path/to/existing/traineddata \
        [--perfect_sample_delay 0] [--debug_interval 0] \
        [--max_iterations 0] [--target_error_rate 0.01] \
        --train_listfile /path/to/list/of/filenames.txt
```

`[i] and [ii] will give the output as` **`model_name.checkpoint`**`.The flag with its default values:`

- `--debug_interval 0` , the trainer outputs a progress report every 100 iterations.
- `--perfect_sample_delay 0`, discards perfect samples if there haven't been that many imperfect ones seen since the last perfect sample. The current default value of zero uses all samples. In practice the value doesn't seem to have a huge effect, and if training is allowed to run long enough, zero produces the best results.
- `--target_error_rate 0.01`, Stop training if the mean percent error rate gets below this value.
- `--max_iterations 0`, Stop training after this many iterations.

Note: All the flags are set to its default value for the whole training.

Combining the output

The lstmtraining program outputs two kinds of checkpoint files:

- `<model_base>_checkpoint` is the latest model file.
- `<model_base><char_error>_<iteration>.checkpoint` is periodically written as the model with the best training error. It is a training dump just like the checkpoint, but is smaller because it doesn't have a backup model to be used if the training runs into divergence.

Either of these files can be converted to a standard traineddata file as follows:

```
training/lstmtraining --stop_training \
  --continue_from ~/tesstutorial/eng_from_chi/base_checkpoint \
  --traineddata ~/tesstutorial/engtrain/eng/eng.traineddata \
  --model_output ~/tesstutorial/eng_from_chi/eng.traineddata
```

This will extract the recognition model from the training dump, and insert it into the --traineddata argument, along with the unicharset, recoder, and any dawgs that were provided during training.

We perform 3 different fine tuning operations using different modified training data and existing models:

sl.no	ben.training_txt	old_traineddata	new_traineddata	error_rate	iteration
1	**[5.1.1]**	**tessdata_best/ ben.traineddata [4.3.1]**	**man_0.traineddata [O-1]**	0.006	89800
2	**[5.1.2]**	**tessdata_best/ ben.traineddata [4.3.1]**	**man_1.traineddata [O-2]**	0.01	125700
3	**[5.1.1]**	**tessdata_best/ script/ Bengali.traineddata [4.3.2]**	**man_2.traineddata [O-3]**	0.01	79200

7. Results and Analysis

The three models **man_0, man_1, man_2** and the existing model **ben** and **Bengali** were tested with the same set of input data.

```
Tesseract input.tiff output[.txt] -l model_name [--psm 3] [--oem 0]
```

page segmentation mode:

--psm 3, fully automatic page segmentation, but no OSD. (Default)

--psm 6, Assume a single uniform block of text.

OCR Engine mode:

--oem 0, legacy mode.

--oem 1, lstm engine.

We prepared a ground truth (.txt) and a pdf for the same, using Ghostscript with size -r720x720 -g6120x7920 the pdf is converted into a .tiff file.

The result (out.txt) generate by tesseract 4.1.0-rc4 with each of the three models are evaluate using the **ocr-evaluation-tools** by UNLV[1]. All the five models that we used are shown, along with the psm, oem, model used (-l), and the accuracy rate of the respective model in the following table.

Model (-l)	Font	psm	oem	Accuracy rate (%)
ben	Lohit_Bengali	3	1	88.59
ben	Lohit_Bengali	6	1	90.86
ben	Bangla_Medium	3	1	90.81
ben	Bangla_Medium	6	1	91.20
Bengali	Lohit_Bengali	3	1	90.92
Bengali	Lohit_Bengali	6	1	93.10
Bengali	Bangla_Medium	3	1	92.41
Bengali	Bangla_Medium	6	1	92.82
man_0	Lohit_Bengali	3	1	90.63
man_0	Lohit_Bengali	6	1	92.99
man_0	Bangla_Medium	3	1	92.63
man_0	Bangla_Medium	6	1	92.76
man_1	Lohit_Bengali	3	1	91.87
man_1	Lohit_Bengali	6	1	94.21
man_1	Bangla_Medium	3	1	94.25
man_1	Bangla_Medium	6	1	94.57
man_2	Lohit_Bengali	3	1	91.71
man_2	Lohit_Bengali	6	1	94.11
man_2	Bangla_Medium	3	1	93.49
man_2	Bangla_Medium	6	1	93.91

Here is a sample of the evaluation report generated by OCR-evaluation-tool.

```
UNLV-ISRI OCR Accuracy Report Version 5.1
----------------------------------------
  53897   Characters
   2926   Errors
  94.57%  Accuracy

      0   Reject Characters
      0   Suspect Markers
      0   False Marks
   0.00%  Characters Marked
  94.57%  Accuracy After Correction

    Ins    Subst     Del   Errors
      0        0       0        0   Marked
    246     1703     977     2926   Unmarked
    246     1703     977     2926   Total

  Count   Missed   %Right
    107      107     0.00   Unassigned
   7717      477    93.82   ASCII Spacing Characters
    465        5    98.92   ASCII Special Symbols
      6        1    83.33   ASCII Digits
      1        1     0.00   ASCII Uppercase Letters
      4        4     0.00   ASCII Lowercase Letters
  45585     1348    97.04   Bengali
     12        6    50.00   General Punctuation
  53897     1949    96.38   Total

 Errors   Marked   Correct-Generated
   1604        0   {য়}-{য়}
    430        0   { }-{<\n>}
     96        0   {}-{}
     57        0   { য়}-{<\n>য়}
     52        0   {আ}-{আ}
     48        0   {ো}-{ো}
     18        0   {ঐ}-{এ}
     18        0   {য়ৰ}-{য়র}
     16        0   {ড়}-{ড়}
     12        0   {ঐ}-{এঁ} //[not EOF]
```

In the report above most of the error count for characters " য়" and " ড়" are due to the misclustering of the characters "য়" as (য ◌), and "ড়" as (ড ◌) and not because of misclassifications by the model. This type of errors are easily fixable during post-processing.

Evaluation reports and input data link here.

Conclusion and Future work

As we can see the newly generated model perform more accurately than the existing Bengali models on the Manipuri words with the model “**man_1**” [--psm 6 and –fontlist Bangla Medium], [input tiff -r300 (300 dpi)] giving the highest accuracy of **94.54%** and with post-processing the accuracy can be increase upto 97.35% (considering the report shown above). Currently, we are working to provide an optimized resolution for the input image, as the accuracy of the OCR model depends on the DPI of the image.