Tesseract OCR for Hindi Typewritten Documents

Jaspreet Kaur
department of Computer Scieance
Punjabi University
Patiala, India
Jaspreet.ptl@gmail.com

Vishal Goyal
department of Computer Science
Punjabi University
Patiala, India
vishal.pup@gmail.com

Manish Kumar
department of Computer Science
Baba Farid College Deon
Bathinda, India
manish21578@gmail.com

2021 Sixth International Conference on Image Information Processing (ICIIP) | 978-1-6654-3361-7/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICIIP53038.2021.9702659

Abstract— The development of Optical Character Recognition (OCR) in Indian text is an active area of research today. The presence of a large number of characters in a set of alphabets, their complex combinations is a major challenge for the OCR designer. This OCR is giving very good results even for Hindi documents that are machine printed. But, it is not giving good results for typewriter typed Hindi documents. Even there is no OCR that is available to date which is capable of recognizing text from typewriter typed Hindi documents. This paper introduces an automated training data framework, provided only with labelled text images, thus removing the need for manually generated text. In contrast to the training images and images text as ground-truth, this approach is based on the random, rule-based generation of meaningless text in an image file and their ground-truth text file. In this paper we describe a dataset of typewriter type Hindi documents and Ground truth (GT) typewritten Hindi text images paired with their transcription. Typewriter typed documents can be incorporated into the repository of Plagiarism detection tools because text cannot be recognized by any OCR. Thus, the functionality of the existing OCR needs to be extended for recognizing typewriter typed documents.

Keywords: Improve accuracy, Optical Character Recognition Software, Tesseract training

I. Introduction

Tesseract is a character recognition system. It was firstly developed at HP in between 1984 to1994. It was modified in 1995 with more accuracy. HP released Tesseract for open source in 2005. It is rigorously tested by developers running through Windows, Linux and Ubuntu. Tesseract up to version 2 could only accept Tiff images with single-column text. These early changes do not include design analysis, multiple columns data, images, or equations into the text resulted in a confusing result. Tesseract supports outgoing text formatting, OCR-related information, and page layout analysis. It is used to convert image text into standard PDF or Word text. It is free, open source software that uses the Command-Line Interface. Tesseract is one of the most accurate open source engines available now. However, because it is open source software, anyone with programming knowledge can edit the code behind Tesseract. It can be operated on Windows, Mac, and Linux machines.

Tesseract is a great general OCR tool that, while trained to recognize text in documents, and is able to work with various problems. Like many other models, it requires that the images be pre-scanned to contain only text – which means that it works best when combined with a text-isolation algorithm. Tesseract OCR is an Open Source OCR. This OCR supports and can recognize more than 100 languages including Hindi. This OCR is giving very good results even for Hindi documents that are machine printed.

III. Working of Tesseract

II. Related work

Ray Smith [5] The Tesseract OCR engine, as was the HP Research Prototype for OCR Accuracy in Fourth Annual UNLV Test. Emphasis is placed on the novel or at least unusual in an OCR engine that includes line detection, classification methods, and classifier. Uwe Springmann et al. [10] this paper presents the basic of German and Latin ground truth (GT) historical data of the OCR in the form of printed text line images with their ground truth transcription. The Special form of GT as pairs of line based image transcription enables you to directly train OCR software recognition models employing recurring neural networks in LSTM architecture such as Tesseract. Ebin Zacharias et al. [11] this paper discusses a critical false-positive case scenario occurred during testing and describes in detail. In comparison to read the text in scanned documents, the intended problem is much more complex. It is helpful to find the text in natural scenes with greater accuracy due to the availability of images below the limits.

Remus Petrescu et al. [12] presents how the different types of OCR engines can be used in the scanned documents for the best result and analyze the potential result of each system based on their performance. Optical Character Recognition is the techniques that are widely used in order to identify the characters in the images obtained after scanning. This method has improved the chances of creating a correct result by combining each result into a single output. Dan Sporici et al. [13] the presented work aims to improve the accuracy of the Tesseract OCR engine by using convolution-based preprocessing kernels. Tesseract has shown excellent performance when tested with appropriate input and its ability to find the characters correctly.

Karishma Verma and Manjeet Singh [15] this paper presented a detailed review of handwriting recognition using convolutional neural network. In computer vision handwriting recognition is a major problem and the main challenge in this field is to deal with enormous variety of handwriting style by the writers. The similarities of the characters make it more difficult to detect the characters.

Jyoti Dalal and Sumiran Daiya [16] in this paper Character recognition techniques are associated with symbolic identity and image of the characters. Each character is located, separated and the resulting character image is inserted into a pre-processor to reduce noise. After classification the identified characters are combined to reconstruct the character of the first character and the context can be used to identify and correct errors.

Tesseract works as a scanner. Its interface is very simple as it requires input with basic commands. We need to input any image.

It takes only two arguments: First is the input image that containing text and the second argument is an output file which is usually text file. Tesseract automatically picks the extension of the output file as text file. There no need to specify the output file extension.

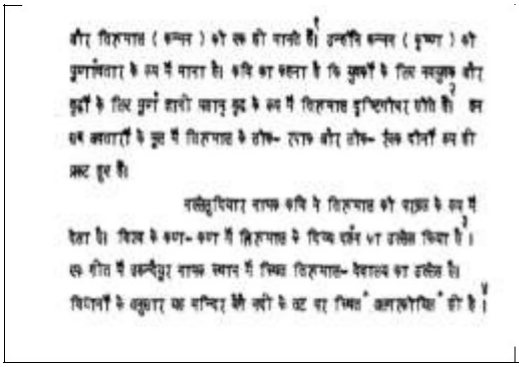

Fig. 1. Input image of typewritten document

Fig. 2 Output of Typewritten image in a text file

A. Create Box file from Image:

Tesseract requires a box file to start the training process with images. The box file is a text file that lists the characters in the training image with the coordinates of the bounding box around the images.

To get the output in the terminal, run the generic command with the path of the given image.

tesseract [image_path] outputfile –l language_name makebox

Image path: This parameter is used to indicate the path of image/name of the image with a extension of image (like jpg, png). Image should be proper.

Output file: This parameter is used to indicate the result of the image after processing a command store with this name.

Language name: this L parameter indicates the language when converting image files with content expect English; choose the language for the OCR engine. To launch OCR engine and is specify some language (like Hindi, Punjabi).

Makebox: The command creates a box containing the specified text. The width of the box is identifying by the optional width argument.

- First number (left)
- Second number (bottom)
- Third number (right)
- Fourth number (top)

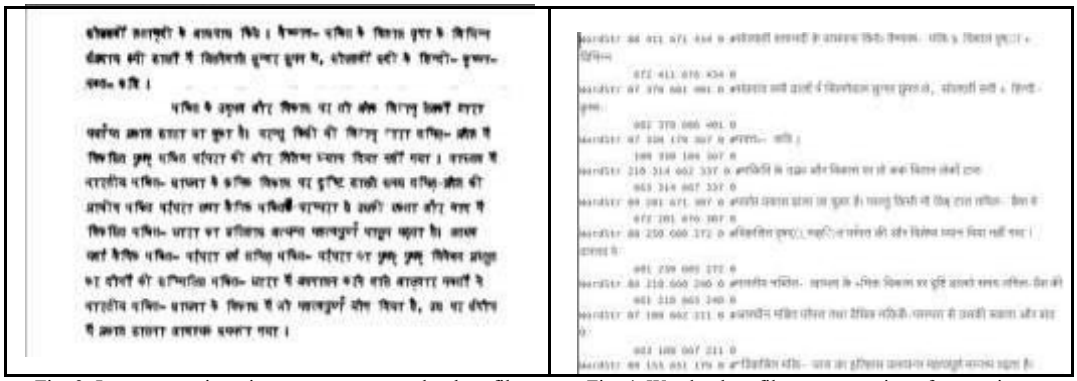

Fig. 3. Input typewritten image to create wordstr box file

Fig. 4. Wordstr box file representation of typewritten Image

B. Correction in Box files

The box file is a text file that lists the characters in the training image. Tesseract 4.0 has a mode where it will extract the text file, but if the character set is different from its current training, it will generate the incorrect text. So the process is to manually edit the file to insert the correct characters into it. This process is done with the help of an image and box file. This is done with the help of Devanagari font. Reading the image and box file character by character if any character in the box file is incorrect then with the help of Hindi font.

Fig. 5 Correction in boxfile of typewritten image

C. LSTM (Long Short Term Memory)

The Lstm training program is a tool for lstm training. Lstm-based networks using a list of lstmf files and starter trained data file as input. Training from scratch is not recommended for users. Fine tuning or other training options

IV. Proposed Method

A. Fine Tuning for Impact Plus Minus

It is the process to train the existing model on new data. But we can add characters to the character set. This process can be performed by using float models that is used for the trained data files. Tesseract base, the complete LSTM model and everything else is required for training is collected in the data file. For Tesseract to identify and can be used. The box files can be converted for use with LSTM training by adding a tab character at end of each line. Then Combine Images with box files into Lstm files. The image and Box files are not being directly give to the trainer, tesseract work with the special file which Combine image, box file and text for each pair of tiff and box file. The samples of Hindi typewritten characters for this paper have been acquired from various Hindi typewriter typed documents.

The next task is to create a Box file containing the required information about each character's binding box in the training image. To produce a set of characters file, Tesseract requires a unicharsetfile.

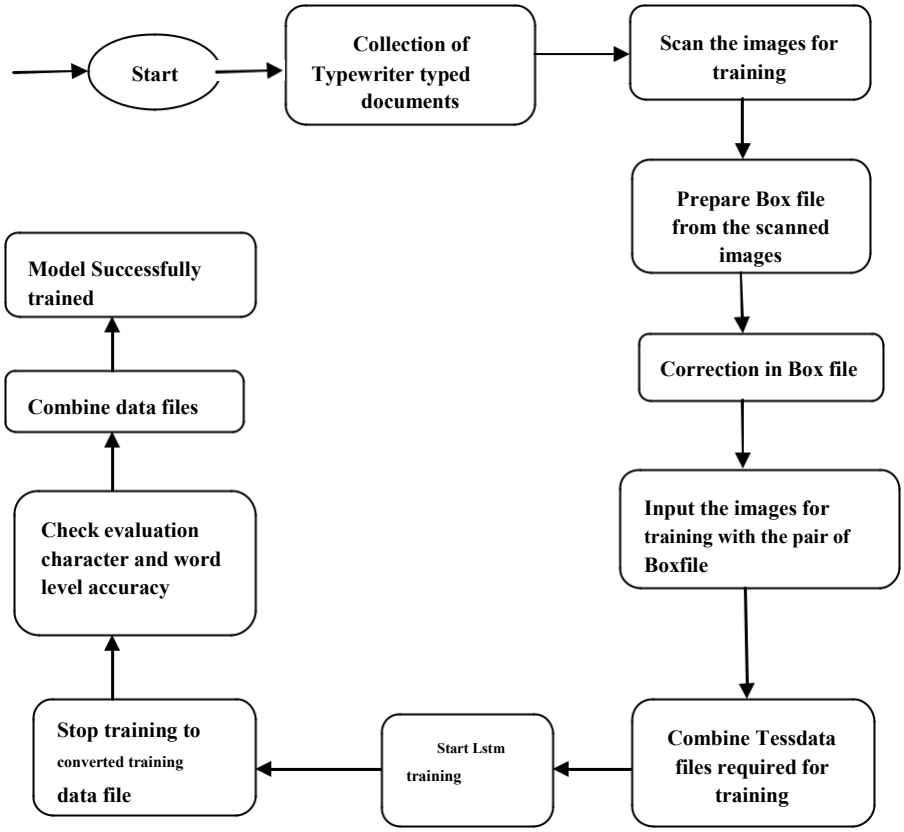

Fig. 6 illustrates the flowchart depicting the adopted research methodology

As Tesseract base, a starter trained data file is provide during the training and has to be set up in advance. The config file is used for providing the control parameters and unicharset file that is used to define the character set.

Algorithm 1: Tesseract Training Procedure
1: procedure: MYPROCEDURE
2: Start training Hindi language, here Hindi language typewritten.
3: input: Typewriter type Hindi documents Dataset
4: top: preprocessing: prepare data set
Scan images for training
5: loop:
6: $tesseract <image file> <box file> makebox: prepare box file
7: $tesseract <image file> <box file> box.train : training each image box file
8: $tesseract<image file> <box file> Correction in box files
9: $tesseract<image file> <box file> sample.gt.txt : Ground truth transcription (GT) for file
10: $lstmtraining <filename.lstmf> unicharset : Prepare lstmf files
11: $unicharset_extractor <box file> : character set file take box file as input
12: $cp existing trained data into new project
13: $combine tessdata –e :extracting tessdata components
14: $lstmtraining :start training max_iterations 400
15: $stop_training :stop training to convert trained data

```
16:   $convert trained data       :check eval char error rate and word error rate
17: end loop
18: $combine_tessdata     : combine generated files
19: output: Successfully Trained
```

B. Training Process

To train Tesseract, we need to copy the trained data file and place it in the tessdata folder in Tesseract. During training, we have to write checkpoint files, that is standard behaviour for neural network trainers. These checkpoints files allow training to be stopped and continued again later when desired. In this case, fine tune of specific training data continue with existing checkpoint file, or in an LSTM model file that has been extracted from an existing file by using combine tessdata. If the unicharset is modified in the trained data flag, as compared to the one that was used in the model provided by using continue from command line flag. This process allows the mapping between the character sets. The training data is provided through lstmf files, which are included Document Data. It contains an image and also the corresponding UTF8 text transcription and can be generated from images and box file pairs.

We obtained an Error rate to be 16.374 after completion of LSTM training. Thereafter, the evaluation of fine-tuned data is initiated for the impact and the execution intended for evaluation of fine-tuned data. Here we are using Tesseract for converting images into text files and completing the LSTM training. In the next step, we have to check the evaluation error rate for word and character level of our model with already existing models.

Table 1 Table depicts the comparative values of actual and modified error rates.

Error rate	Actual	Modified
Char Error Rate	29.352948	11.516595
Word Error Rate	60.127517	29.110729

Table 2 displays the character level accuracy report obtained from tesseract Devanagari trained data. Table 2 depicts the comparative values of actual and modified error rates.

Accuracy report Characters accuracy	Actual	Modified
Characters	24106	24106
Errors	2616	1877
Accuracy %	89.15%	92.12%

Table 2. Defines LSTM training results and comparison of actual results with modified results. This step initiate with

LSTM training to check the ocreval character level accuracy. In the table results shows obtained by using Devanagari trained data exists trained data file to check the accuracy of ocreval characters level. There are a total of 24106 characters, 2616 errors, and 89.15% accuracy as shown in the table for Actual. In the case of Modified results obtained by using hin trained data is new trained data file created during training file to check the accuracy of ocreval characters level. There are a total of 24106 characters, 1877 errors, and 92.12% accuracy as shown in the table for Modified.

Table 3 displays the word level accuracy report obtained from tesseract Devanagari trained data.
Table depicts the comparative values of actual and modified error rates

Accuracy report Words accuracy	Actual	Modified
Words	4396	4396
Misrecognized	1261	1011
Accuracy %	71.31%	77.00%

Table 3. Defines LSTM training results and comparison of actual results with modified results. This step initiate with LSTM training to check the ocreval word level accuracy. In the table results shows obtained by using Devanagri.traineddata (existing trained data) file to check the accuracy of ocreval characters level. There are a total of 4396 words, 1261 misrecognized, and 71.31% accuracy as shown in the table for Actual. In the case of Modified results obtained by using hin trained data (new trained data file created during training) file to check the accuracy of ocreval characters level. There are a total of 4396 words, 1261 misrecognized, and 77.00% accuracy as shown in the table for Modified.

V. Conclusions

In this, an open source engine Tesseract is used for recognition of Hindi typewritten documents. The biggest advantage of Tesseract OCR is its availability as an open source code. Therefore, anyone who has an interest in learning the process and the ability to improve it, can be trained in a new language. In this case, the step by step process of training the Tesseract engine for Hindi typewriter typed text document has been explained. Then the training procedure of Tesseract is done to recognize the Hindi typewritten documents, and the results were observed. It has been found that editing the box file manually is a cumbersome task. we tried to generate the box file automatically.

VI. Future Scope

OCR is a very useful and popular application. The idea of developing and improving the OCR for Punjabi text is a very different aspect. The experiment could be used to build translation and increase efficiency to read and fetch the type-writer words form an image and any text book also , it can also be used in other language scripts, as well as in handwritten characters. Developing commercial OCR

systems that can maintain high levels of recognition regardless of the irregularities such as the low quality of the input documents, and different font styles are a challenge for Hindi and other Indian scripts. Indian languages have many additional challenges such as a set of capital letters due to editors, lack of general test knowledge, and their lack of support from browsers, operating system, and keyboard, etc. Punjabi character segmentation poses additional problems due to touching, and overlapping characters.

References

[1] El Gajoui, Khadija , A. Allah, and Fadou, "Optical Character Recognition for Multilingual Documents: Amazighe-French," The 2nd World Conference on Complex Systems, Agadir, Morocco, November 2014.

[2] A. Belaïd, "Reconnaissance automatique de l'écriture et du document, Campus scientifique," 2001.

[3] C. Patel., V. Harish, M. Swathi, CH. Deepthi, "A Review on the Various Techniques used for Optical Character Recognition," International Journal of Engineering Research and Applications, vol. 2, pp.659-662, jan-feb 2012.

[4] G. Gabrani, A. Solomon, U. Dviwedi, "Handwritten Statement Analysis Using Neural Networks," International conference on Signal Processing, Communication, Power and Embedded System (SCOPES), vol. 5, Issue 9, pp. 308-317,2016.

[5] R. Smith, "An Overview of the Tesseract OCR Engine," Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), vol. 2, pp. 629-633, 2007.

[6] C. Adak, B. Chaudhuri, M. Blumenstein, "Legibility and Aesthetic Analysis of Handwriting," 14th IAPR International Conference on Document Analysis and Recognition, vol. 4, Issue 13, pp. 481-489, 2017.

[7] R. Verma, M. Raghuwanshi, "Comparative Study of Various Techniques on Handwriting Recognition and Analysis," 3rd International Conference for Convergence in Technology (I2CT), Vol. 22, Issue 8, pp. 1930-1936, 2018.

[8] M. Luiz, M. Tamazin, M. Khedr, "Robust Arabic Handwritten Word Recognition Technique Based on Wavelet Transform", 8th International Conference on Computer Science and Information Technology (CSIT), vol. 41, Issue 19, pp.164-175, 2018.

[9] H. Cao, R. Prasad, and P. Natarajan, "Handwritten and Typewritten Text Identification and Recognition using Hidden Markov Models," International Conference on Document Analysis and Recognition, vol. 31, Issue 15, pp.259-268, 2011.

[10] U. Springmann, C. Reul, S. Dipper, and J. Baiter, "Ground Truth for training OCR engines on historical documents in German Fraktur and Early Modern Latin," pp. 1-19, 2020.

[11] E. Zacharias, M. Teuchler and B. Bernier, "Image Processing Based Scene-Text Detection and Recognition with Tesseract," pp. 1-6, 2020.

[12] R. Petrescu, S. Manolache, C. Boiangiu, G. Vlasceanu, C. Avatavului, M. Prodan, and I. Bucur, "Combining Tesseract and Asprise Results to Imrpove OCR Text Detection Accuracy", Journal of Information Systems Management, vol. 13, Issue 01, pp. 57-64, 2019.

[13] D. Sporici, E. Cus ,and C. Boiangiu, "Improving the Accuracy of Tesseract 4.0 OCR Engine Using Convolution-Based Preprocessing," Symmetry, vol.12, pp. 715-732, 2020.

[14] I. Pal, M. Rajani, A. Poojary, and P. Prasad, "Implementation of Image to Text Conversion using Android App," International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, vol. 6, Issue 4, pp.2291-2297, 2017.

[15] K. Verma and M. Singh, "A Literature Survey on Handwritten Character Recognition using Convolutional Neural Network," International Journal of Management, Technology And Engineering, vol. 8, Issue 7, pp. 257-263, 2018.

[16] J. Dalal and S. Daiya, "Image Processing based Optical Character Recognition using Matlab," International Journal of Engineering Sciences and Research Technology, pp. 406-411, 2018.

[17] Neha and D. Ahlawat, "Handwritten Alphanumeric character recognition and comparison of classification techniques," International Journal of Engineering Sciences and Research Technology, pp. 419-428, 2018.

[18] Madhura, and M. Kumar, "Enhanced Image Transferring Scheme using Security Techniques," International Journal of Computer Sciences and Engineering, vol. 6, Issue 7, pp. 446-451, 2018.

[19] S. Pletschacher, J. Hu and A. Antonacopoulos, "A New Framework for Recognition of Heavily Degraded Characters in Historical Typewritten Documents Based on Semi-Supervised Clustering," 10th International Conference on Document Analysis and Recognition, vol. 37, Issue 26, pp. 910-917, 2009.

[20] N. Abubacker, R. Gandhi, "An Extended Method for Recognition of Broken Typewritten Characters Special Reference to Tamil Script," IEEE Conference on Open Systems (ICOS2011), vol. 13, Issue 51, pp. 259-268, 2011.

[21] G. Retsinas, B. Gatos, A. Antonacopoulos, G. Louloudis and N. Stamatopoulos, "Historical Typewritten Document Recognition Using Minimal User Interaction," ACM, vol. 5, Issue 18, pp. 1045-1052, 2015.

[22] A. Mahmood, A. Srivastava, "A Novel Segmentation Technique for Urdu Typewritten Text," Recent Advances on Engineering, Technology and Computational Sciences (RAETCS), vol. 64, Issue 27, pp. 118-129, 2018.

[23] S. Panda, J. Tripath, "Odia Offline Typewritten Character Recognition using Template Matchingwith Unicode Mapping," International Symposium on Advanced Computing and Communication (ISACC), vol. 37, Issue 27, pp. 442-449, 2015.