A Review of Usage of Tesseract OCR Engine with Vernacular Indian Languages

Kartik Joshi(✉) and Harshal Arolkar

Faculty of Computer Applications and Information Technology, GLS University, Ahmedabad, India
kartikjoshi@ymail.com

Abstract. Optical Character Recognition, or OCR, is a tool that could apprehend textual content in photographs or scanned documents and convert it into system-readable textual content. To recognize and extract information from snap shots, optical character popularity (OCR) software program uses exclusive codes and codes to research the form, sample and grouping of characters. In 2005, Hewlett-Packard (HP) created the open supply software program Tesseract OCR (Optical person reputation). Its reason is to extract text from images and convert it into gadget-readable text. Tesseract can manage a couple of fonts, sizes and languages, making it clean to apply for OCR. In this newsletter, the authors provide an in-intensity evaluate of the usage of the Tesseract OCR engine in Indian languages.

Keywords: Optical Character Recognition · Indian languages · Tesseract OCR engine

1 Introduction

Tesseract is a nicely-reviewed open-source OCR engine advanced by Hewlett Packard (HP) between 1984 and 1994. Tesseract started as a PhD research assignment at HP Laboratories in Bristol and quickly won traction as a potential software program and/or hardware upload directly to HP's flatbed printer variety. This momentum is because of the reality that the OCR engine was still in its infancy at that point and could not emphasize anything however the maximum pressure. In reality, Tesseract predated the engine commercial enterprise after a merger among the groups in Colorado, however, it never has become a commercial product. The following section of its improvement took place at HP Bristol Laboratories as compressed OCR studies. This work makes a specialty of improving extraction performance in place of baseline accuracy. Development of the assignment resulted in 1994. HP released Tesseract as open source in late 2005. Now available at http://code.google.com/p/tesseract-ocr [1].

2 Overview of Tesseract OCR

The special steps inside the Tesseract OCR procedure are proven in the Fig. (1). All steps noted are to output textual content statistics from entering photograph data.

The operating principle of the Tesseract OCR engine can be summarized as follows:

P. Pareek et al. (Eds.): IC4S 2024, LNICST 599, pp. 3–10, 2025.
https://doi.org/10.1007/978-3-031-77081-4_1

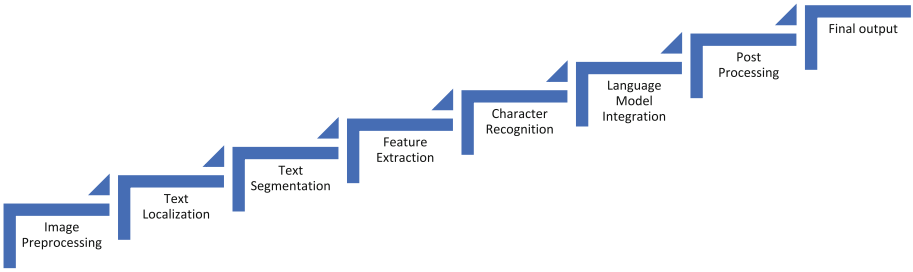

Fig. 1. Generic steps to derive a text file from an input image file

1. Picture Pre-Processing:- Pre-processing is required to enhance the quality and prepare the entered photo for OCR. These may additionally consist of resizing, binarization (changing the photo to black and white), noise discount, and different upgrades that grow accuracy.
2. Textual content localization: Tesseract unearths areas of the photo that need to include textual content. This procedure calls for identifying areas containing text and separating them for further analysis.
3. Textual content Segmentation:- text is segmented into phrases or characters. Approval of each character is dependent on completing those steps.
4. Function extraction: Tesseract collects features of every segmented phrase or character. Those functions assist the OCR engine in recognizing the fine of every individual, together with the form, length, and distribution of pixels.
5. Feature:- Use the extracted item to compare the character with the getting-to-know version. Tesseract uses neural networks and other device-learning algorithms to understand characters. Because the OCR engine is skilled in distinct documents, it may understand characters in distinct fonts and styles.
6. Integrating language models:- Tesseract integrates language fashions to boost accuracy and help multiple languages. Language patterns provide context and commands to the OCR engine, permitting the engine to make better decisions about the textual content it recognizes.
7. Put up-processing:- The OCR engine performs post-processing after person recognition to accurate errors and improves usual textual content popularity. Writing evaluations, content material-primarily based modifying, and different methods to enhance the accuracy of the output could be a part of this.
8. Output: - The device-readable textual content extracted from the output picture is the final output of the Tesseract OCR engine. This output can be used for a ramification of programs, along with text extraction and digitization, or can be processed, listed, and so forth. Variables consisting of complex textual content and words can affect the accuracy of your outcomes. To get the first-rate results, users need to set parameters or specify specific parameters for particular packages or languages [2]

Because of its initial launch in 2005, Tesseract OCR has long passed through many modifications and upgrades version. Due to its accuracy, pace, and overall performance on antique and new neural networks (NN), Tesseract has three forms of LSTM models for maximum languages. This version can be utilized by specifying the “OEM” command

line. The LSTM layer improves reputation accuracy using modeling the long-time period dependence of collection signals.

The general purpose of LSTM is to recognize continuous textual content, together with character dependencies and content material within a word or line. LSTMs are useful for textual content recognition in photographs where the order of attributes is essential because they can pick out long-term dependencies in sequences [2, 3].

Tesseract OCR now helps Hindi, Bengali, Telugu, Tamil, Kannada, and many others. It supports many Indian languages together with. Indian languages and their developers are education fashions and developing engines for a higher reputation of textual content and emblems in these languages. Extraordinary Indian languages get hold of unique tiers of guides under popularity.

While working with an OCR engine, you frequently need to provide the necessary data documents to understand Indian languages and specify code phrases with the use of Tesseract OCR. Profiles and schooling models are blanketed in the records document for each supported language. Set suitable parameters in the API for Tesseract as a library:

```
tesseract input_image.jpg output_text.txt -l guj
```

In this example, "-l guj" specifies the Gujarati language for OCR. You can replace "guj" with the language code corresponding to the Indian language you are working with [4].

Please note that OCR accuracy for Indian languages can also vary depending on the first class of the input image, the accuracy of the text, and the availability of Tesseract language-specific information. Tesseract can also upload new language fashions to improve the rendering of text in input photos after education.

3 Literature Review

There is a substantial body of literature available for the recognition of English, Japanese, Chinese, and Arabian characters, but only a small amount of work has been reported for the recognition of Indian scripts. Consonants, vowels, script representation, and conjunctional appearance vary greatly across Indian scripts. Because of these differences, developing accurate character recognition becomes a huge challenge.

Jaspreet Kaur *et al.* in [5] have worked on Improving the accuracy of the Tesseract OCR engine for machine-printed Hindi documents. The advancement of Optical Character Recognition (OCR) of Indian text is a current area of study but the presence of numerous characters in a set of alphabets, as well as their intricate combinations and intricate grapes, pose a challenge to the OCR engine. The purpose of this paper is to improve the efficiency of the Tesseract OCR in Hindi. Google's Tesseract Optical Character Recognition software is introduced in this paper. The current work aims to improve the Tesseract 4.0 OCR engine's accuracy.

N. Mishra *et al.* in [6], states that the Tesseract OCR engine is said to be one of the most important open-source OCR engines nowadays. Tesseract OCR can recognize Hindi as of version 3.01, however it nonetheless needs a few enhancements to enhance overall performance. Regardless of the printed text, the reputation of Hindi may be very low because hyphenated characters in Hindi can not be without problems separated due

to partial overlap. They concluded that their scheme solved this trouble and allowed the Tesseract OCR engine to effortlessly section and apprehend Devanagari harmony symbols. This article also compares Devanagari OCR engines primarily based on reputation, runtime, font conversion, and library size.

M. A. Hasnat *et al.* in [7], tried to combine text reputation into Tesseract OCR. They say they represent the system of schooling the Tesseract OCR engine for a new language version, Bengali, by using developing documents. They concluded that with sufficiently big training records, Tesseract OCR might be trained to improve the overall performance of existing languages or create entirely new languages.

Partha Chakraborty *et al.* in [8], researchers diagnosed meaningful words and terms from pics based totally on the OCR Tesseract engine and NLTK. They show how the Tesseract works in Bengali and finish what it approaches to understand the textual content in the NLTK. The proposed OCR method accepts a photograph as input and converts it into searchable facts. The system can also search for words in the generated textual content and show Bengali-based content material. First, it detects phrases and contours, then it analyzes the words, then it separates the characters with static conduct, then it performs the analysis, and finally the transformation. it's far a framework that includes OCR in addition to an excellent language gadget to categorize content with Bengali meaning within the outputs.

Milind Kumar Audichya *et al.* in [9], researchers laboured on recognizing Gujarati characters through the usage of Tesseract OCR. In this newsletter, they represent how Teserract works. They used Tesseract OCR education records collectively to digitally recognize characters in images. They tested various font styles, types, and sizes and discovered fantastic consequences. They concluded that Teserract does now not produce exact outcomes when coping with complex characters or similar characters.

N. Desai *et al.* in [10], have worked on Printed Gujarati Character Recognition: A Review. This publication provides an overview of several OCR approaches. Pre-processing, segmentation, characteristic extraction, classification, and publish-processing are the steps in the OCR pipeline. OCR gadgets, registration code recognition, smart libraries, etc. could be used in lots of real-time packages inclusive notwithstanding extensive OCR research, person popularity in regional languages which include Arabic, Sindhi, and Urdu remains a challenge. They determined that Gujarati character statistics records may be used on Kaggle and advanced in one-of-a-kind codecs and sizes. The effects were inconclusive because the information they analyzed did now not follow big characters or semi-written characters in Gujarati.

B. Kataria *et al.* in [11], researchers studied the personal reputation of Sanskrit textual content based on CNN bidirectional LSTM: a complete literature review and methodology. They concluded that OCR may be very tough to increase in Indian languages because of the inclusion of various alphabets, complex characters, and variables. Systems evolved in recent years do not necessarily produce accurate outcomes for complicated languages like Gujarati. This paper provides two stepped-forward CNNs for character recognition of written characters and numbers written in Gujarati. The results of deep gaining knowledge are as compared with traditional techniques and some primary techniques utilized by specific authors on similar materials (such as Sanskrit and Bengali). As compared to different present algorithms, the accuracy of the CNN-percent model is accelerated

by using a mean of 18.29%. The second proposed CNN-HGC version accomplished 7.60% and 14.6% accuracy as compared to different current algorithms for handwriting Gujarati numbers and symbols, respectively. Notwithstanding large and diverse statistics, the proposed CNN architecture achieves a clean typical result compared to other rules and methods inside the literature.

Mamata Nayak *et al.* in [12], have worked on Odia Characters Recognition by Training Tesseract OCR Engine. They have represented the process of training the Tesseract OCR engine for an entirely new language model, Odia by creating a dataset. They concluded that with a substantially large amount of training dataset, Tesseract OCR can be trained for an existing language to improve its efficiency or create an entirely new model.

Suraya Mubeen *et al.* in [13], They represent Tesseract OCR's work on existing Tamil language models. They concluded that the design may want to seize and extract textual content from pictures. But, the Tesseract engine is the most accurate; moreover, the output is stored in a text file and copied to the report. Tesseract works nicely whilst the image follows these regulations: separation of foreground and history, horizontal alignment and appropriate scaling, correct photograph pleasant, blur, and noise. The primary downside of this model is that the image is blurry or bright.

Nisal Udawatta *et al.* in [14], researchers established that the Tesseract OCR engine can create Sinhala report files and way to use the statistics with custom policies to perceive Sinhala characters in scanned documents. An OCR engine is often used to create textual content files. When Sinhala language records are used within the OCR characteristic, it returns text recognition, confidence popularity, and the position of the text within the authentic picture. The accuracy of the system can be checked by thinking about the reliability of every phrase. The magnificence turned into skilled use of 40 characters with 20 photos every and tested with over one thousand characters (two hundred in keeping with the phrase) in distinct font sizes with about ninety percent accuracy. Each line longer than 20 phrases takes much less than 0.05 s; this is a great improvement over the guide facts entry. A classifier can be designed for energetic studying due to the fact it can reuse check snapshots.

Chamila Liyanage *et al.* in [15], to broaden business-grade Tamil OCR for font and text length. They offered the development of Tamil optical characters for Tamil announcements. Given that those three letters are closely related to each other, the techniques used to generate OCR for Bengali and Sinhala can without problems be carried out to Tamil letters as well. The satisfactory mixture from the essay layout is a mixture of 3 fonts and three sizes. The outcomes had been compared with present Tamil mods in Tesseract for evaluation. They concluded that their system was carried out with 81% accuracy, 12.5% higher than the existing Tesseract Tamil module. it can be concluded that the accuracy of the Tesseract engine is based on records and phrases. Languages like Hindi, Sanskrit, Odiya, and Tamil require extra schooling than Gujarati and Bengali, subsequently the above-referred languages. This removes the need to create files with special fonts and sizes for Gujarati and Bengali languages to improve the overall performance of the version.

The comparative analysis of the literature review is given in Table 1.

From the above table, it can be concluded that the accuracy of Tesseract Engine is dataset and language-dependent. Languages such as Hindi, Sanskrit, Odia, and Tamil

Table 1. Comparative Analysis of papers reviewed

Paper Sr. No.	Dataset Language	Character dataset type	Dataset size in pages	Source of dataset	Accuracy achieved
4	Hindi	Alpha-Numeric	1000	Books	85%
5	Hindi	Alpha-Numeric	1000	Books	85%
6	Bengali	Alpha-Numeric	3200	Books, Newspapers, Journals	98%
7	Bengali	Alpha-Numeric	3750	Self-created	68%
8	Gujarati	Alphabets	100	Books, Documents, Newspapers	80%
9	Gujarati	Alphabets	385	Kaggle	Not Available
10	Sanskrit	Alphabets	1000	Self-created	92%
11	Odia	Alpha-Numeric	Not Available	Books, Journals	96%
12	Tamil	Alpha-Numeric	1000	Self-created	93%
13	Sinhala	Alpha-Numeric	18000	Self-created	86%
14	Tamil	Alpha-Numeric	1000	Self-created	81%

have been trained more as compared to Gujarati and Bengali, due to which the above efficiency is generated. This summarises the need to create a dataset for Gujarati and Bengali languages for different types of fonts and sizes to improve the overall efficiency of the model.

4 Observations

Although Tesseract has established benefits in lots of situations, it has a few barriers:

1. Difficulty with complex techniques and systems: Tesseract can also have issues processing documents with complex strategies, complicated systems, or more than one column. it can be misinterpreted or unusual with unusual spellings.
2. Writing textual content: Tesseract is particularly used for printing and has restricted capability for textual content. Handwriting varies from individual to person, making it tough for the OCR engine to apprehend exceptional handwriting.
3. Low-nice pics: Tesseract works satisfactorily with snapshots with crisp, clear textual content. Low decision or blurriness may bring about reduced high-quality and consequently decreased accuracy.
4. Noisy images: Noisy pictures inclusive of history styles, watermarks, or artifacts can lessen Tesseract's accuracy. Pre-processing may also require cleaning the picture earlier than feeding it to the OCR engine.

5. Language aid: Tesseract supports many languages, however, performance varies from language to language. a few languages might also have extra aid and accuracy than others.
6. Language aid: Tesseract helps many languages, however its overall performance varies from language to language. a few languages may have more help and accuracy than others. Understanding the content material is not sufficient: Tesseract makes snapshots that the character symbolizes and may not apprehend the content of the text. This may be deceptive in cases where this means relies upon the order of phrases or sentences.
7. Constrained help for non-Latin alphabets: Tesseract is designed for Latin alphabets, and although it has been prolonged to support different alphabets, its overall performance on non-Latin alphabets won't be consistent.
8. Constrained language processing (NLP) skills: Tesseract focuses on textual content recognition and extraction however cannot flawlessly procedure words. He can not apprehend the means and significance of the textual content he produces.
9. Installation and Setup: even though Tesseract may be installed, the setup method may be time-consuming and difficult. The OCR engine wishes to be great-tuned generally to acquire the first-rate overall performance. Its effectiveness depends on the statistics used, the pleasantness of the input image, and appropriate preprocessing.

5 Conclusion

In this paper, researchers were capable of presenting information approximately the performance of the Tesseract OCR engine for numerous Indian languages. Based totally on the evaluation of the literature, it can be concluded that although researchers have made efforts to expand OCR primarily based on the popularity of diverse Indian languages, they have not yet completed the preferred results on all characters. A character, code, and pattern result are promising, but validating the person appears to leave loads of room for development.

References

1. Tesseract OCR GitHub. https://github.com/tesseract-ocr. Accessed 14 Jan 2024
2. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8) (1997)
3. Smith, R.: An overview of the tesseract OCR engine. In: Ninth International Conference on Document Analysis and Recognition, Parana, vol. 2, pp. 629–633. IEEE (2007)
4. Patel, C., Patel, A., Patel, D.: Optical character recognition by open-source OCR tool tesseract: a case study. Int. J. Comput. Appl. **55**, 50–56 (2012)
5. Kaur, J., Goyal, V., Kumar, M.: Improving the accuracy of tesseract OCR engine for machine printed Hindi documents. In: AIP Conference Proceedings, vol. 2455. AIP, Chennai (2022)
6. Mishra, N., Patvardhan, C., Lakshimi, V. C., Singh, S.: Shirorekha chopping integrated tesseract OCR engine for enhanced Hindi language recognition. Int. J. Comput. Appl. **39**(6), 19–23 (2012)
7. Hasnat, M.A., Chowdhury, M.R., Khan, M.: Integrating Bangla script recognition support in Tesseract OCR. BRAC University (2009)

8. Chakraborty, P., et al.: Recognize meaningful words and idioms from the images based on OCR tesseract engine and NLTK. In: Gupta, D., Goswami, R.S., Banerjee, S., Tanveer, M., Pachori, R.B. (eds.) Pattern Recognition and Data Analysis with Applications. LNEE, vol. 888, pp. 297–310. Springer, Singapore (2022). https://doi.org/10.1007/978-981-19-1520-8_23
9. Audichya, M., Saini, J.: A Study to recognize printed Gujarati characters using tesseract OCR. Int. J. Res. Appl. Sci. Eng. Technol. (IJRASET) **5**(9), 1505–1510 (2017)
10. Desai, N., Hasan, M., Swadas, P.: Printed Gujarati character recognition: a review. Int. Res. J. Eng. Technol. (IRJET) **9**(4) (2022)
11. Kataria, B., Jethva, H.B.: CNN-bidirectional LSTM based optical character recognition of Sanskrit manuscripts: a comprehensive systematic literature review. Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol. (IJSRCSEIT) **5**(2), 2456–3307 (2019)
12. Nayak, M., Kumar, A.: Odia characters recognition by training tesseract OCR engine. Int. J. Comput. Appl. (ICDCIT) **1**, 25–30 (2013)
13. Mubeen, S., Brahmani, J., Kalyan, D., Jagirdar, A., Kumar, A.: Optical character recognition using tesseract. Int. J. Res. Appl. Sci. Eng. Technol. (IJRASET) **10**, 672–675 (2022)
14. Udawatta, N., Liyanage, S.: Sinhala character recognition using Tesseract OCR. In: The Third International Conference on Advances in Computing and Technology (ICACT) (2018)
15. Liyanage, C., Nadungodage, T., Weerasinghe, R.: Developing a commercial grade Tamil OCR for recognizing font and size-independent text. In: Fifteenth International Conference on Advances in ICT for Emerging Regions (ICTer) (2015)