Enhancing Hindi OCR Accuracy with Large Language Model-Based Post-Processing

Sreevidya B S[1], Manjula S[2], Dr. Shobha T[3], Gauravesh Sharma[4]

[1]Assistant Professor, Department of CSE, BMSCE

[2]Assistant Professor, Department of CSE, BMSCE

[3]Assistant Professor, Department of ISE, BMSCE

[4]Undergraduate Student, Department of ISE, BMSCE

Abstract

Optical Character Recognition (OCR) is a technology that fetches text present in images - generally documents. However, the emergence of multiple fonts to make text appealing has made it increasingly difficult for Optical Character Recognition (OCR) to take place with high precision. This is especially true for Hindi where “robust and efficient recognizers are not yet commercially available” according to Venu G. in his book “Guide to OCR for Indic Scripts”. Teja K. and Jyoti P. attempted to solve this problem in their paper titled "Multi-font Devanagari Text Recognition”, however a key issue of word error was not addressed. In a paper by Ray Smith, Tesseract OCR for Hindi was found to have a word error rate of 69.44 percent and character error of 15.41 percent. This paper implements Tesseract OCR for Hindi, trained on multiple Hindi fonts, and incorporates pre-processing techniques such as de-skewing, noise reduction, and binarization. The further investigation has been done for exploring a Large Language Model (LLM)—specifically GPT-4—that can correct recognition errors produced by the standard Hindi Tesseract engine. Using a 125-page corpus comprising Mahabharata commentary, epics, and modern newspapers, applied a lightweight image pre-processing pipeline, performed baseline OCR, and feed the raw output to the LLM for token-level correction. The integrated pipeline reduces the CER to 2.47% and the WER to 5.83% on held-out data.

Keywords— Optical Character Recognition (OCR), Large Language Models (LLM), Pre-processing, Post-processing, Word Error Rate.

I. INTRODUCTION

Optical character recognition (OCR) is a technology that allows a user to detect the text present in an image. Adobe describes the same as “a technology that changes printed documents into digital image files” [1]. The OCR technology uses machine learning algorithms such as Neural Networks, Support Vector Machines (SVM), Convolutional neural networks (ConvNet) among others for text recognition [2]. Furthermore, OCR can be refined by adding ‘pre-processing’ and ‘post-processing’, the former deals with refining the image before entering into the OCR, while the latter focuses on correcting small mistakes made by the model.

Hence, OCR can be used in a variety of sectors including (1) Education, (2) Banking, (3) Research (4) Logistics. [1][2][3]. Although a much-debated topic, the technology was initially

conceptualized between the late 19th century to the early 20th century [4], but only towards the 1970s that it can be said with certainty that Ray Kurzweil, the founder of Kurzweil Computer Products became the first to build and implement an OCR model which according to sources (like Adobe and others) “ [The

model] could recognize text that was printed in just about any font."[1]. The company founded by Ray was later acquired by Xerox and since then OCR has been able to increase its accuracy – especially for Roman script.

While early research focused on multi-font adaptability [5], recent advancements stress the importance of post-processing to enhance OCR accuracy. Post-processing techniques, particularly those powered by language models may potentially help correct recognition errors by using linguistic context, which is especially valuable for visually complex scripts.

Yet the growth of OCR for other scripts has been relatively slow compared to Roman scripts. One such script type is Devanagari which is used for languages such as Hindi, Marathi, Sanskrit, as mentioned in a book titled "Guide to OCR for Indic Scripts" by Venu G. it is highlighted that "for Roman scripts, we have commercial OCRs which can produce reasonably accurate text, for a wide variety of documents. However, robust and efficient recognizers are not yet commercially available for Indian languages." [6]. More than a decade later, we have developed commercial OCRs for Hindi, such as Tesseract, Indsenz OCR [7], and eAksharayan OCR [8]. These OCRs are still not as efficient in Devanagari as in Roman scripts; nonetheless, they showcase increased awareness. Tesseract was selected for this paper's implementation due to its ease of training and being open source despite having a large word error rate as demonstrated in a study by Ray Smith [9] – demonstrating a word error rate of 69.44 percent and character error rate of 15.41 percent.

II. RELATED WORK

Early rule-based efforts to recognise printed Devanagari date back to the 1970s. Sinha and Mahabala introduced a primitive-search recogniser that matched strokes against stored structural templates, coining many of the

feature descriptors still used today [10], [11]. Soon after, Sethi and Chatterjee proposed a four-stage pipeline—global preprocessing, shirorekha (headline) removal, feature extraction, and decision logic based on D-curves, C-curves and stroke slants—to improve recognition speed and modularity[11].

Major leaps in OCR, however, can be seen from the 2010s. Such as Chirag I Patel highlighting a method to scan documents using Artificial Neural Network [12] and Dileep Kumar Patel in his paper utilizing discrete wavelet transform (DWT) and Euclidean distance metric (EDM) [13]. Advancements were seen regarding segmentation and processing.

More recent work has turned to sequence models. Teja K. leveraged Long Short-Term Memory (LSTM) networks to create a Multifont Devanagari OCR system, but omitted word-error-rate reporting and offered limited analysis of post-recognition noise [14]. Meanwhile, large-scale benchmarks for English and French show that most residual errors stem not from glyph classification but from missing diacritics, broken ligatures and contextual spelling—all of which can be mitigated by language-aware post-processing.

Large Language Models (LLMs) offer a principled way to perform that correction. Thomas et al. demonstrated that GPT-3 could cut word-error rate by an order of magnitude when cleaning OCR output from historical English newspapers [15]. To date, however, no peer-reviewed study has applied an LLM-driven post-processing layer to Hindi or any other Devanagari-based language, leaving an open research gap that the present work addresses.

III. METHODOLOGY

This paper implements three-stage pipeline—image pre- processing, OCR recognition, and LLM-based post- processing. The project begins with the preprocessing stage of the image where first we apply techniques to improve the raw image. Next the image is sent to an OCR engine and finally the image is sent to the post processing stage.

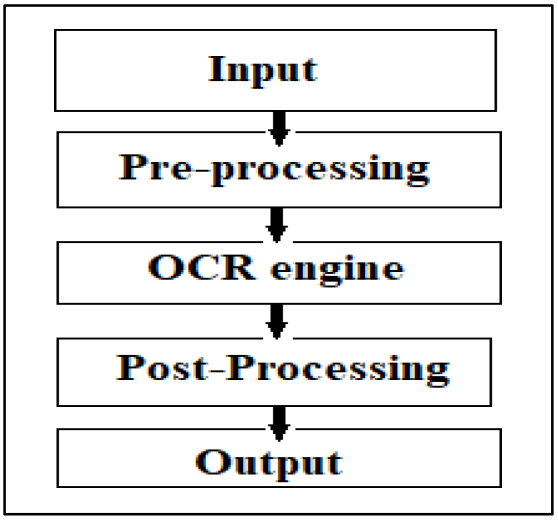

Figure 1: Steps in the proposed pipeline

A. *Preprocessing*

Preprocessing is the first step of creating an OCR model. This involves using multiple techniques to enhance the image quality. The most popular language to do this is python and the libraries include OpenCV and Pillow. For this project we used techniques such as grayscale, binarization, skew correction and noise reduction.

a) Grayscale:

This process converts an image from other color spaces to shades of Gray. We utilized OpenCV and the command cv2.cvtColor(image, cv2.COLOR_BGR2GRAY) to achieve the same.

b) Binarization

This step converts any colored image into a binary image that contains only vtwo colors black and white. This makes text identification much easier. Auto binarization is the advanced version of the same with the difference being automatic determination of threshold value.

c) Skew Correction

The process of skew correction (De-skewing) is straightening the text present in a slightly tilted format. This paper utilized a technique by Leo Ertuna using OpenCV [16].

d) Noise Removal

The process of noise removal takes place by using two major techniques. First being erosion, which is removal of small spots in the image, followed by dilation which is thickening the text which might have become thin due to erosion.

B. *OCR Engine*

The next step involves sending the pre-processed image to the OCR engine. The OCR used for this project is Tesseract [17] due to its ease of training nature. For this project we further used Viet OCR [18] along with JtessBox to set the ground truth. Fonts with low baseline accuracy—*Himalaya*, *Devangi Bold*, and *Ritu*—were retrained with additional synthetic glyphs. Open-source fonts from [19] were used for evaluation.

C. *Post-Processing*

Post processing in OCR consists of two steps error detection and error correction. Most OCR systems use dictionary-based error correction. However, for this paper an emerging technology namely Large Language Model was utilized.

Large Language Models are generative models which predict the upcoming text from pretrained vectors. IBM defines the same as a "category of foundation models trained on immense amounts of data making them capable of understanding and generating natural language and other types of content to perform a wide range of tasks." [20]

This paper utilized Chat GPT-4 for creating a post-processor. To achieve the same custom GPT was implemented along with the multiple prompts to to reduce word-error rate (WER) and character-error rate (CER). The final post-processor deployment is also available at the GPT store which available in the GPT Store (public link in [21]). The prompt to implement the same is also listed below.

"The GPT is designed to act as a post-processor for OCR (Optical Character Recognition) output, specializing in both English and Hindi texts. It receives text input that may contain various errors such as incorrect spacing, typos, incorrect paragraph divisions, segmentation errors, and other mistakes common in OCR outputs. Its main role is to identify and correct these errors, refining the text into its correct, readable form for both languages. The GPT focuses on ensuring accuracy and coherence in the text, making it suitable for further use or analysis. It pays special attention to common OCR errors, applying corrections to spacing, spelling, paragraph structure, and segmentation to enhance the overall quality of the text.

The GPT avoids making assumptions about the content of the text, focusing solely on structural and grammatical accuracy in both English and Hindi. It corrects the input without adding any additional commentary or asking for clarification, regardless of the text's clarity. This approach ensures a streamlined correction process, emphasizing efficiency and precision in addressing OCR text errors in both languages."

III. DATASET

For evaluation we curated a 125-page Devanagari validation corpus drawn from two public-domain sources. Eighty pages originate from the Bhandarkar Oriental Institute's critical edition of the Mahābhārata, and forty-five pages come from the June 2021 archive of the Hindi daily Dainik Bhaskar. Every scan is an 8-bit, 300 dpi TIFF created from the institutions' master microfilm or PDF files. Blurred or duplicate images—about two per cent of the raw collection— were discarded.

The remaining pages were deskewed, cropped to the text region, and page-aligned with UTF-8 transcriptions supplied by the curators; manual spot-checks on ten per cent of pages confirmed zero offset errors. Because our recognizer was fine-tuned on separate synthetic glyph data, the entire 125-page corpus is reserved exclusively for validation and metric reporting.

	Collection	Pages	Genre	Licence
S1	Mahabharatha	80	Epic Scripture	Public Domain
S2	Dainik Bhaskar archive	45	News	CC-BY-NC 4.0

Table I. Composition of the 125-page validation corpus

IV. TESTING AND RESULTS

For testing the project two main criteria were utilized which were stated in an article by Leung titled " Evaluate OCR Output Quality"[22]. The two criteria being character error rate and word error rate. All 125 validation pages were included in the evaluation.

A. Character Error Rate (CER)

The character error rate is defined as the number of substitution, deletion and insertion errors found per document and is given by the formula where S is the number of substitutions, D is the number of deletions, and I is the number of insertions and finally N is the number of characters present in the document. See figure 2 for the corresponding formula. The final character error rate was found to be 2.47 percent compared to the original 15.41 percent.

$$CER = \frac{S + D + I}{N}$$

Fig 2: Character Error Rate Formula

B. Word Error Rate (WER)

The word error rate is defined as the number of substitution, deletion, and inserting errors found with respect to words per document. The difference is that a misinterpreted word will count as a complete error. See figure 3 for the corresponding formula. The word error rate found in this paper was 5.83 percent compared to the original 69.44 percent.

$$WER = \frac{S_w + D_w + I_w}{N_w}$$

Fig 3: Word Error Rate Formula

As summarised in Fig. 4, both CER and WER drop sharply at each successive stage of the pipeline.

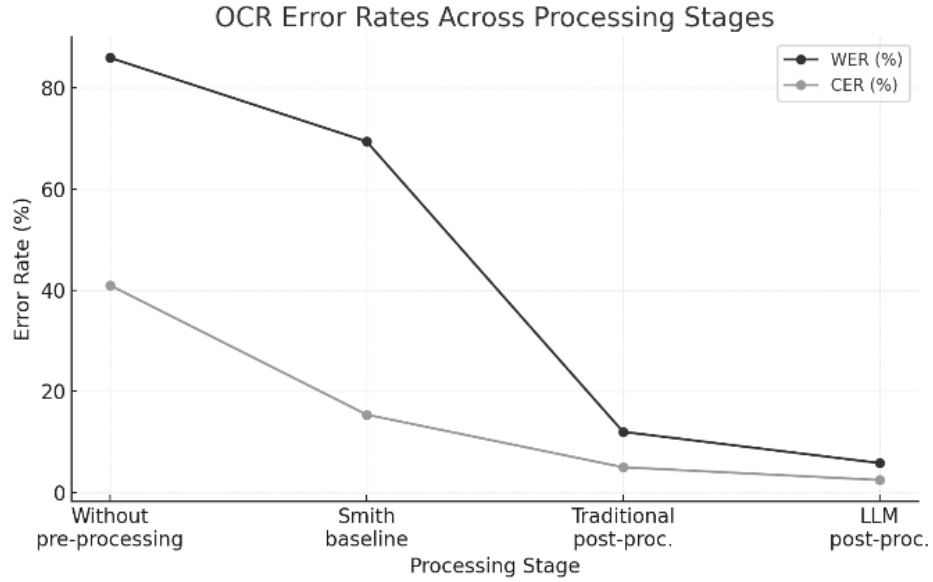

Fig 4: OCR error rates across different post-processing

V. CONCLUSION.

This paper presents a Hindi OCR pipeline that combines image pre-processing, a fine-tuned Tesseract recognizer, and GPT-4 post-processing. The model had the required steps of preprocessing by applying features such as deskewing, noise

reduction and binarization. The post processing was done using Chat GPT-4 configuration. Finally, the OCR was implemented using Tesseract. The Final model achieved a word rate error of 5.83 percent and char error rate of 2.47 percent.

Future improvements include utilizing open-source large language models and providing specific training instead of relying on configuration.

REFERENCES

[1]. Adobe. "OCR: How It Works and Why It Matters." *Adobe.com*, www.adobe.com/acrobat/guides/what-is-ocr.html.

[2]. Hamad, Karez, and Mehmet Kaya. "A Detailed Analysis of Optical Character Recognition Technology." *International Journal of Applied Mathematics, Electronics and Computers*, 2016, https://doi.org/10.18100/ijamec.270374.

[3]. Amazon Web Services. "What Is OCR (Optical Character Recognition)?"

[4]. *AWS.com*, aws.amazon.com/what-is/ocr/.

[5]. Everen, Dave V. "The History of OCR." *Veryfi Blog*, 28 Feb 2023, www.veryfi.com/ocr-api-platform/history-of-ocr.

[6]. Sheppard, Shannon M., et al. "One Font Doesn't Fit All." *Education Sciences*, vol. 13, 2023, p. 864, https://doi.org/10.3390/educsci13090864.

[7]. Govindaraju, Venu, and Srirangaraj R. Setlur. *Guide to OCR for Indic Scripts*. Springer,

2010, https://doi.org/10.1007/978-1-84800-330-9.

[8]. Indsenz OCR. www.indsenz.com/int/index.php.

[9]. eAksharayan OCR. tdil-dc.in/eocr/index.html.

[10]. Smith, Ray, Rika Antonova, and Dar-Shyang Lee. "Adapting the Tesseract Open Source OCR Engine for Multilingual OCR." *Proceedings of the 2009 International Conference on Document Analysis and Recognition*, 2009, 10.1145/1577802.1577804.

[11]. "Hindi Fonts Download." Easy Nepali Typing, www.easynepalityping.com/download-hindi-fonts.

[12]. IBM. "What Are Large Language Models (LLMs)?" IBM, 2 Jan. 2024, www.ibm.com/topics/large-language-models.

[13]. "Post Processor" Fixes OCR text errors, including segmentation, in English and Hindi. GPTs, Open AI, 4 Apr. 2024. https://chatgpt.com/g/g-tFDKzj7as-post-processor.

[14]. Sinha, R. M. K., and H. N. Mahabala. "Machine Recognition of Devanagari Script." *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, 1979, pp. 435-441.

[15]. Sethi, K., and B. Chatterjee. "Machine Recognition of Constrained Hand-Printed Devnagari Numerals." *Journal of the Institution of Electronics and Telecommunication Engineers*, vol. 22, 1976, pp. 532-535.

[16]. Patel, Chirag I., R. Patel, and P. Patel. "Handwritten Character Recognition Using Neural Network." *International Journal of Scientific & Engineering Research*, vol. 2, no. 5, May 2011.

[17]. Patel, Dileep K., et al. "Handwritten Character Recognition Using Multi-Resolution Technique and Euclidean Distance Metric." *Journal of Signal and Information Processing*, 2012.

[18]. Kundaikar, Teja, and Jyoti D. Pawar. "Multi-Font Devanagari Text Recognition Using LSTM Neural Networks." *Proceedings of the 1st International Conference on Sustainable Technologies for Computational Intelligence*, 2018, pp. 495-506, https://doi.org/10.1007/978-981-15-0029- 9_39.

[19]. Thomas, Alan, et al. "Leveraging LLMs for Post-OCR Correction of Historical Newspapers." *Proceedings of the Third Workshop on Language Technologies for Historical and Ancient Languages (LT4HALA)*, 2024, pp. 116-121, https://aclanthology.org/2024.lt4hala-1.14/.

[20]. Ertuna, Leo. "How to Automatically Deskew (Straighten) a Text Image Using OpenCV." *Becoming Human: Artificial Intelligence Magazine*, 2020, https://becominghuman.ai/how-to-automatically-deskew-straighten-a-text- image-using-opencv-a0c30aed83df.

[21]. Tesseract-OCR Project. "Fonts for Tesseract Training." *tesseract- ocr.github.io*, tesseract-ocr.github.io/tessdoc/Fonts.html.

[22]. Nguyen, Q. "VietOCR." *SourceForge* and *GitHub*, 3 Dec 2024, sourceforge.net/projects/vietocr/, https://github.com/nguyenq/VietOCR3.

[23]. "Download Hindi Fonts." *Easy Nepali Typing*, www.easynepalityping.com/download-hindi-fonts.

[24]. IBM. "What Are Large Language Models (LLMs)?" *IBM.com*, 2 Jan 2024, www.ibm.com/topics/large-language-models.

[25]. OpenAI GPT Store. "Post Processor – Fixes OCR Text Errors in English and Hindi." 4 Apr 2024, https://chatgpt.com/g/g-tFDKzj7as-post-processor.

[26]. Leung, Kenneth. "Evaluate OCR Output Quality with Character Error Rate (CER) and Word Error Rate (WER)." *Towards Data Science*, 24 Jan 2021, towardsdatascience.com/evaluating-ocr-output-quality-with- character-error-rate-cer-and-word-error-rate-wer-853175297510.