Zero-shot OCR Accuracy of Low-Resourced Languages: A Comparative Analysis on Sinhala and Tamil

Nevidu Jayatilleke and **Nisansa de Silva**
Department of Computer Science & Engineering,
University of Moratuwa, Sri Lanka
{nevidu.25, NisansaDdS}@cse.mrt.ac.lk

Abstract

Solving the problem of Optical Character Recognition (OCR) on printed text for Latin and its derivative scripts can now be considered settled due to the volumes of research done on English and other High-Resourced Languages (HRL). However, for Low-Resourced Languages (LRL) that use unique scripts, it remains an open problem. This study presents a comparative analysis of the zero-shot performance of six distinct OCR engines on two LRLs: Sinhala and Tamil. The selected engines include both commercial and open-source systems, aiming to evaluate the strengths of each category. The Cloud Vision API, Surya, Document AI, and Tesseract were evaluated for both Sinhala and Tamil, while Subasa OCR and EasyOCR were examined for only one language due to their limitations. The performance of these systems was rigorously analysed using five measurement techniques to assess accuracy at both the character and word levels. According to the findings, Surya delivered the best performance for Sinhala across all metrics, with a WER of 2.61%. Conversely, Document AI excelled across all metrics for Tamil, highlighted by a very low CER of 0.78%. In addition to the above analysis, we also introduce a novel synthetic Tamil OCR benchmarking dataset[1].

1 Introduction

Optical Character Recognition (OCR) is a computational technology that is used for recognising text within digital images, such as scanned documents, advertisements, and photographs (Agarwal and Anastasopoulos, 2024; Jain et al., 2021; Weerasinghe et al., 2008). OCR is commonly employed as an information entry tool to extract valuable data from scanned documents such as forms, receipts, invoices, and passports. Historically, OCR (along with Text-To-Speech systems) was created to assist blind or disabled individuals by facilitating machines to read written text aloud to them, a development that dates back to 1914 (Mittal and Garg, 2020).

The process of OCR typically involves multiple steps: 1) It begins with image acquisition, where the image is captured. 2) Next is preprocessing, which enhances the image quality and includes binarisation to separate the content from the background. 3) Following this is layout analysis, where the document is divided into distinct regions. 4) The next step is character-level segmentation, which breaks the text down into lines, words, and individual characters. 5) Recognition follows, involving feature extraction and classification to identify the characters. 6) Finally, post-processing improves the results, often using language models. Each of these stages is essential for effective OCR performance (Jain et al., 2021; Nazeem et al., 2024).

While OCR systems have advanced significantly, particularly for High-Resourced Languages (HRL) such as English and French (Nazeem et al., 2024), recognising text from complex or low-quality images, historical documents, and Low-Resource Languages (LRL) still presents challenges (Agarwal and Anastasopoulos, 2024). In this benchmarking study, we conduct a thorough examination of various multilingual and monolingual OCR systems, assessing their capabilities for two selected low-resource languages in South Asia: Sinhala and Tamil.

Sinhala is an Indo-European language spoken as L1 by just 16 million people, mostly located in the island of Sri Lanka (de Silva, 2025). Sinhala has a script that is unique to it which descends from the Indian Brahmi Script (Fernando, 1949). Tamil is a Dravidian language spoken as L1 by around 79 million people located primarily in India, Sri Lanka,

[1] https://huggingface.co/datasets/Nevidu/tamil_synthetic_ocr

Proceedings of Recent Advances in Natural Language Processing, pages 471–480
Varna, Sep 8–10, 2025
https://doi.org/10.26615/978-954-452-098-4-056

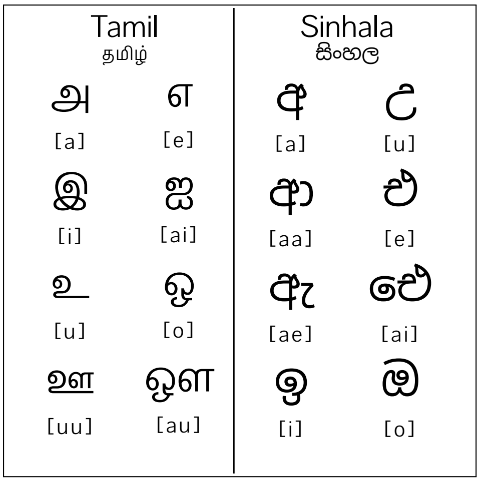


Figure 1: An example of the use of rounded script in Tamil and Sinhala languages.

and Singapore (Wijeratne et al., 2019). It also has a unique script that is also a descendant of the Indian Brahmi Script (Paneerselvam, 1972). Both Sinhala and Tamil are considered LRLs by the criteria proposed by Ranathunga and de Silva (2022), where Sinhala is deemed to be lower resourced (Category 02) as opposed to Tamil (Category 03).

2 Existing Works

Despite extensive research over the past several decades, the challenge of accurately recognising Sinhala characters in OCR systems remains a formidable obstacle (Anuradha et al., 2020). Indic languages, such as Tamil, present a myriad of complexities and character variations, significantly complicating the development of effective OCR solutions. In clear terms, the accuracy of South Asian rounded scripts is significantly behind that of Latin-based scripts, highlighting a crucial area for improvement and development (Anuradha et al., 2021).

2.1 Sinhala OCR Systems

Several research studies have been conducted on developing Sinhala OCR systems. A proposed system for the Sinhala language by Anuradha et al. (2020) utilises the Tesseract 4.0 OCR engine with a graphical user interface, comprising a user, an API, the Tesseract engine (Smith, 2007), a post-processor, and a data store. The Tesseract engine employs Long Short Term Memory (LSTM) based deep learning techniques for text recognition from images. However, it occasionally fails to recognise certain characters. The post-processor addresses this by applying linguistic rules for improved accuracy. Using various commercially available font types, varying in size, this system achieved an average accuracy of 94%.

A study on multi-style printed Sinhala character recognition (Maduranga and Jayalal, 2022) utilized a hybrid Artificial Neural Network (ANN) model, following four steps: Data Preprocessing for image enhancement and noise removal; Feature Extraction, dividing 50x50px character images into 9 zones with 12 pieces each to create a 108-signal feature vector; Development and Training, using line features from an 850-character database (mainly Iskoola Pota font[2]) to train a backpropagation network in MATLAB, achieving about 75% training accuracy over 138 epochs; and Testing, evaluating performance with a dataset of 1253 characters.

Velayuthan and Ambegoda (2025) conducted a comparative analysis of OCR models, including Surya[3], TR-OCR[4], EasyOCR[5], and Tesseract OCR (Smith, 2007), focusing on digitising documents in low-resource languages. Although the study aimed at Sinhala and Tamil, it reported results only for Sinhala using synthetic datasets and for English via the FUNSD dataset (Jaume et al., 2019). Metrics like CER (Character Error Rate) and WER (Word Error Rate) were employed, revealing that Surya outperformed the other models on the Sinhala datasets. While all models had high error rates on English, Surya was the most balanced for Sinhala, achieving good accuracy with moderate computational demands and superior power efficiency, using 0.69 kWh less on average than TR-OCR. The accuracy difference between Sinhala and English is attributed to the nature of the datasets, with synthetic Sinhala datasets versus noisy English form document images.

2.2 Tamil OCR Systems

Various research initiatives have been undertaken to create OCR systems for the Tamil language. Unlike Sinhala, which is primarily spoken in Sri Lanka, the Tamil language has a much wider geographical spread across the South Asian region. Notwithstanding their different linguistic roots, Tamil also utilises a rounded script similar to Sinhala, as depicted in Figure 1, due to their writing systems

[2] https://learn.microsoft.com/en-us/typography/font-list/iskoola-pota
[3] https://github.com/VikParuchuri/surya
[4] https://huggingface.co/Ransaka/TrOCR-Sinhala
[5] https://github.com/JaidedAI/EasyOCR

being related. This leads to similar challenges in the development of OCR technology.

Liyanage et al. (2015) developed a Tamil OCR system using the open-source Tesseract OCR engine, inspired by its applications with scripts such as Sinhala and Bangla. The methodology involved creating a 169-character OCR alphabet and preparing training data from selected words in various Unicode fonts. They tested different training combinations and found that a model using data from three fonts at three sizes achieved the best results. The system was evaluated using 20 scanned images from ancient Tamil books, achieving an accuracy of 81%. This was benchmarked only against Tesseract's existing Tamil module, over which, a 12.5% improvement was shown.

Recent research introduced the *Nayana framework* (Kolavi et al., 2025), which enhances Vision-Language Models (VLMs) like GOT (General OCR Theory) (Wei et al., 2024) for low-resource languages, including Tamil. It addresses data scarcity through a layout-aware synthetic data generation pipeline and Low-Rank Adaptation (LoRA) (Hu et al., 2022). This system translates English documents into Tamil while maintaining their layout, followed by a two-phase Cross-Modal Alignment training with LoRA. Nayana-OCR achieved a WER of 0.551 and a METEOR score of 0.592, significantly outperforming the base GOT model (WER 1.020, METEOR 0.051) and other traditional OCR systems like Tesseract (Smith, 2007) and PaddleOCR[6] on the Tamil test set.

3 Methodology

As discussed earlier, contemporary studies are utilising open-source tools to effectively fine-tune models for new languages and improve existing capabilities. Many of these tools offer multilingual support. Additionally, several organisations have developed commercial engines that excel in OCR. In this study, we thoroughly evaluate the capabilities of selected OCR engines for Sinhala and Tamil languages in a zero-shot setting.

3.1 Overview of Selected OCR Technologies

In this study, we benchmark the capabilities of six selected open-source and commercial engines specifically designed for OCR tasks.

Cloud Vision API[7]: The Cloud Vision API enables developers to seamlessly incorporate vision detection features into their applications. This includes functionalities such as image labelling, face and landmark detection, OCR, and the tagging of explicit content. The first version of the API launched for general availability in May 2017. The API is designed to perform OCR on files such as Portable Document Formats (PDFs) and Tag Image File Formats (TIFFs), as well as on images with dense text. It is particularly optimised for image documents containing large amounts of text and those that feature handwriting, allowing for accurate recognition and conversion to machine-readable text.

Document AI[8]: Document AI is a document understanding platform that transforms unstructured data from documents into structured data, making it easier to understand, analyse, and utilise. It employs machine learning and Google Cloud to develop scalable, end-to-end cloud-based document processing applications. The API provides organisation through content classification, entity extraction, advanced searching, and more. The OCR processor specifically enables the identification and extraction of text, including handwritten text, from documents in over 200 languages. Additionally, the processor uses machine learning to assess the quality of a document based on the readability of its content.

Tesseract: Tesseract OCR is a widely used open-source text recognition engine developed by Hewlett-Packard and now funded by Google. It combines Hidden Markov Models and various machine learning algorithms with traditional computer vision techniques. Tesseract 4.0 introduced deep learning methods using LSTM networks, significantly enhancing performance compared to earlier versions that primarily relied on conventional approaches (Smith, 2007; Nazeem et al., 2024). In this study, we use Tesseract 5.5.0, which improves upon 4.0 with the existing LSTM engine and additional performance enhancements.

Subasa OCR[9]: The study by Anuradha et al. (2020) extended the analysis of Sinhala OCR using deep learning (LSTM) with Tesseract 4.0, examining factors such as text genre, image resolution, and algorithmic complexity. Training utilised the UCSC 10M Sinhala corpus[10], incorporating vari-

[6] https://github.com/PaddlePaddle/PaddleOCR

[7] https://cloud.google.com/vision/docs/ocr

[8] https://cloud.google.com/document-ai/

[9] https://ocr.subasa.lk/

[10] https://ltrl.ucsc.lk/tools-and-resourses/

ous fonts and image qualities, with a focus on character segmentation. Evaluation involved 30 test images from old newspapers (200 DPI), old books (72 DPI), and contemporary books (300 DPI), including low-DPI (96px) contemporary images. The OCR engine achieved character accuracy of up to 67.02% on old newspapers, 87.53% on old books, and 87.63% on contemporary books, maintaining a high accuracy of 87.88% on low-DPI images (Anuradha et al., 2021).

Surya[3]: This OCR toolkit supports over 90 languages and benchmarks favourably against cloud services. It offers line-level text detection, layout analysis (including detection of tables, images, headers, etc.), reading order detection, and LaTeX OCR capabilities. The text detection model, built from the ground up using a modified EfficientViT architecture, was trained for three days on four A6000 GPUs. The text recognition model was trained on the same hardware for two weeks, using a modified Donut model that incorporates Grouped Query Attention (GQA) (Ainslie et al., 2023) and a Mixture of Experts (MoE) layer (Shazeer et al., 2017), Unicode Transformation Format-16 (UTF-16) decoding, and changes to the layer configuration. It is important to note that this system is designed for printed text and not for handwriting.

EasyOCR[5]: This is an OCR technique that supports over 80 languages. EasyOCR utilises ResNet (He et al., 2016), LSTM, and Connectionist Temporal Classification (CTC) (Graves et al., 2006) models for character recognition. The detection component of EasyOCR employs the Character Region Awareness For Text detection (CRAFT) Algorithm (Baek et al., 2019). EasyOCR consists of three key elements. The first is feature extraction, which is executed by the ResNet model. The second element is sequence labelling, for which the LSTM algorithm is utilised, and the last component is decoding which relies on CTC. The EasyOCR's `Readtext` function is utilized during text recognition which can read letters and numbers from images while providing their location coordinates (Awalgaonkar et al., 2021).

The Google Cloud Vision API and Document AI are commercial engines, whereas Google Tesseract, Surya, and EasyOCR are open-source systems. Additionally, we selected Subasa OCR, which is a fine-tuned version of the Tesseract model available through a web application[9], though the source code and model are not directly accessible.

3.2 Dataset Selection and Assembly

As noted by Ranathunga et al. (2024); de Silva (2025), free and open datasets for this task are scarce for low-resourced languages; even when some published research may exist. To achieve optimal results, we employed distinct datasets for each of the two selected languages, ensuring a tailored approach that enhances the effectiveness of our analysis. For the Sinhala language, we chose a dataset published in Hugging Face, `sinhala_synthetic_ocr-large`[11] by Ravihara (2024), consisting of 6,969 pairs of images and reference texts created using five different font families; Noto Sans Sinhala[12], Gemunu Libre[13], Noto Serif Sinhala[14], Yaldevi[15], and Abhaya Libre[16].

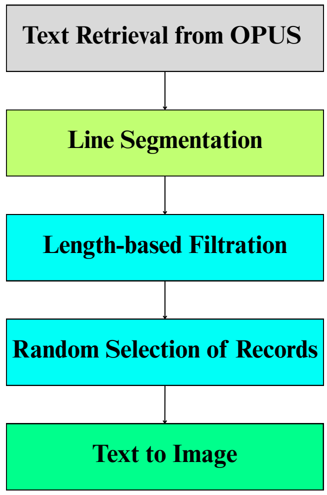


Figure 2: Overview of the Tamil synthetic OCR dataset creation.

Since we considered a synthetically generated dataset for Sinhala, we also aimed to evaluate synthetically generated data for Tamil to ensure a fair comparison. However, we could not find any publicly available Tamil datasets developed in a similar manner to the selected Sinhala dataset. As a result, we decided to create a new dataset for the Tamil language. The overview of the Tamil dataset creation is shown in Figure 2.

The Tamil text was obtained from

[11] https://huggingface.co/datasets/Ransaka/sinhala_synthetic_ocr-large
[12] https://fonts.google.com/noto/specimen/Noto+Sans+Sinhala
[13] https://fonts.google.com/specimen/Gemunu+Libre
[14] https://fonts.google.com/noto/specimen/Noto+Serif+Sinhala
[15] https://fonts.google.com/specimen/Yaldevi
[16] https://fonts.google.com/specimen/Abhaya+Libre

OPUS[17] (Tiedemann, 2012), specifically selecting OpenSubtitles (Lison and Tiedemann, 2016) v2024[18]. The content was then divided line by line, focusing solely on Tamil characters since the primary goal was language evaluation, resulting in 2,437,960 records. Subsequently, this set was filtered to retain only texts longer than 40 characters, yielding 222,658 records. This filtration step ensured that word-level evaluation was accurate. However, to ensure a fair comparison with the Sinhala dataset, we decided to randomly select 7,000 samples from the remaining texts to equalise the sample sizes. This step was necessary to prevent evaluation scores from being skewed by differences in dataset size. Further, conducting OCR on additional samples is challenging due to the resource consumption involved.

We carefully selected six unique font families out of a total of 17 from Google Fonts[19] to diversify the creation of images from the text records, ensuring that each visual representation is impactful for the analysis. We identified the unique fonts visually, as some fonts, such as Mukta Malar[20] and Baloo Thambi 2[21], have characters that can appear very similar to the naked eye. The selected six fonts are Hind Madurai[22], Noto Serif Tamil[23], Kavivanar[24], Noto Sans Tamil[25], Pavanam[26], and Anek Tamil[27].

A function was then developed utilising the capabilities of the `Pillow`[28] library to systematically convert textual data into image files. Its fundamental purpose is to ensure a proportional distribution of input text records across the defined set of font files, promoting an equitable use of fonts in the generated image dataset. For each text entry, the function dynamically calculates optimal image dimensions based on the measurements of the text bounding box, rendering the text in black on a white background. A significant feature of this implementation is the precise centring of the text within the generated image, achieved through calculated positioning in relation to the height and width of the image, thereby enhancing visual consistency and quality.

The post-processing phase for the Tamil dataset involved excluding records without string values for both reference and generated features. For the Sinhala dataset, an additional step removed all non-Sinhala characters from the texts to focus exclusively on the system's targeted linguistic capabilities. Since we created the Tamil dataset, this character-focused post-processing was addressed during preprocessing. This synthetic Tamil OCR public dataset[1] is a key contribution to our study. A few examples of a Tamil sentence in selected fonts are shown in Figure 3.

Figure 3: Examples of a single Tamil sentence in the selected fonts, showing the style differences.

3.3 Integration of OCR Systems

Both the Google Cloud Vision API and Document AI are available through the Google Cloud Platform (GCP). However, while enabling the Cloud Vision API is straightforward, as it only requires activating the API in GCP, Document AI necessitates the manual creation of a processor within GCP. This created processor is then used to initiate the OCR process.

The process of integrating the Tesseract engine was quite simple through the use of the `pytesseract`[29] wrapper. This enabled the automation of character recognition for every record

[17] https://opus.nlpl.eu/
[18] https://opus.nlpl.eu/OpenSubtitles/ta&en/v2024/OpenSubtitles
[19] https://fonts.google.com/
[20] https://fonts.google.com/specimen/Mukta+Malar
[21] https://fonts.google.com/specimen/Baloo+Thambi+2
[22] https://fonts.google.com/specimen/Hind+Madurai
[23] https://fonts.google.com/noto/specimen/Noto+Serif+Tamil
[24] https://fonts.google.com/specimen/Kavivanar
[25] https://fonts.google.com/noto/specimen/Noto+Sans+Tamil
[26] https://fonts.google.com/specimen/Pavanam
[27] https://fonts.google.com/specimen/Anek+Tamil
[28] https://pillow.readthedocs.io/
[29] https://pypi.org/project/pytesseract/

within each dataset, streamlining the overall data processing workflow.

As previously mentioned, the Subasa OCR engine is exclusively accessible via a web application. This limitation meant that we had to manually input images one by one to perform OCR, a process that proves impractical given our substantial Sinhala dataset of 6,969 records. To streamline this tedious task and enhance efficiency, we resolved to automate it using `Selenium`[30] after meticulously identifying the element locations by inspecting the web source code.

The seamless integration of Surya and EasyOCR was straightforward, largely due to their comprehensive documentation. Both engines are accessible on GitHub and can be conveniently installed as libraries directly through Python, making the setup process efficient and user-friendly.

3.4 Evaluation Mechanism

The evaluation was performed using the generated text against the reference text of the datasets. For the comparison, we employed five different measurements;

Character Error Rate (CER): It is based on the concept of Levenshtein distance, which measures the minimum number of character-level operations (substitutions, deletions, and insertions) necessary to transform the ground truth text into the output generated by OCR. The CER formula is expressed as $(S+I+D)/N$, where S represents the number of substitutions, I denotes insertions, D signifies deletions, and N is the total number of characters in the ground truth text (Nazeem et al., 2024).

Word Error Rate (WER): Similar to the CER, the WER is calculated by comparing the text produced by the OCR system against a ground truth or reference text. The WER is determined by the number of word-level errors made by the OCR engine. The formula for calculating the word error rate is also $(S+I+D)/N$, but considered at the word level instead of the character level (Nazeem et al., 2024).

Bilingual Evaluation Understudy (BLEU): It is a method for automatically evaluating machine translation quality by comparing it to one or more professional human translations. The score measures proximity to these human reference translations using a weighted average of variable-length phrase matches, based on modified n-gram precision. It also includes a brevity penalty to discourage overly brief candidate translations. The final BLEU score, ranging from 0 to 1, is calculated as the geometric mean of the modified n-gram precisions multiplied by the brevity penalty (Papineni et al., 2002).

Average Normalised Levenshtein Similarity (ANLS): This metric takes into account both reasoning errors and shortcomings of OCR. To evaluate answers, it calculates a similarity score between the response of the model and the ground truth using Levenshtein distance. A key feature of this scoring system is its application of a threshold (τ=0.5) on the normalised Levenshtein distance (NL): if NL is less than or equal to 0.5, the similarity score is calculated as 1-NL; otherwise, the score is 0. This approach allows ANLS to provide intermediate scores (ranging from 0.5 to 1) for responses that are logically correct but may contain minor recognition errors, contrasting with standard accuracy metrics, which would score them as zero (Biten et al., 2019).

$$\text{ANLS} = \frac{1}{N}\sum_{i=0}^{N}\left(\max_{j} s(a_{ij}, o_{q_i})\right) \tag{1}$$

$$s(a_{ij}, o_{q_i}) = \begin{cases} 1 - NL(a_{ij}, o_{q_i}) & \text{if } NL(a_{ij}, o_{q_i}) < \tau \\ 0 & \text{if } NL(a_{ij}, o_{q_i}) \geq \tau \end{cases}$$

Metric for Evaluation of Translation with Explicit ORdering (METEOR): Similar to BLEU, METEOR is also a metric designed for evaluating the quality of machine translation. It identifies unigram matches between machine-generated outputs and human translations, considering surface forms, stemmed forms, and synonyms. Designed to overcome BLEU's limitations, METEOR computes a score by integrating unigram precision, unigram recall (with greater emphasis on recall), and a fragmentation penalty for word order of matched words. This method has shown better correlation with human judgments and has been widely adopted in OCR studies, making it a preferred choice for evaluation (Banerjee and Lavie, 2005).

4 Discussion of Results

This section analyses the results from our comparative evaluation of OCR systems, presented separately for Sinhala and Tamil in Table 1. Except for Subasa OCR and EasyOCR, all other systems processed both languages selected for evaluation in this study. Subasa OCR is a monolingual system

[30] https://www.selenium.dev/

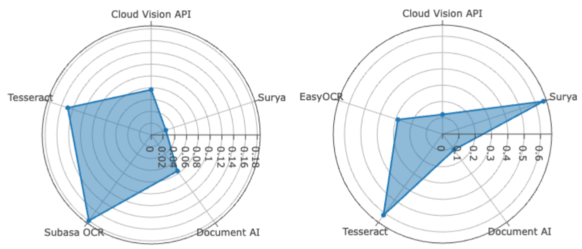
OCR System	Language	CER↓	WER↓	BLEU↑	ANLS↑	METEOR↑
Cloud Vision API	Sinhala	0.0619	0.0767	0.9193	0.9447	0.9269
	Tamil	0.0079	0.1204	0.5790	0.9922	0.8751
Surya	Sinhala	**0.0076**	**0.0261**	**0.9396**	**0.9920**	**0.9723**
	Tamil	0.1392	0.6500	0.1487	0.8672	0.3359
Document AI	Sinhala	0.0610	0.0758	0.9199	0.9455	0.9278
	Tamil	**0.0078**	**0.1198**	**0.5803**	**0.9923**	**0.8762**
Subasa OCR	Sinhala	0.0761	0.1799	0.6894	0.9259	0.8099
	Tamil	-	-	-	-	-
Tesseract	Sinhala	0.0702	0.1489	0.7553	0.9319	0.8436
	Tamil	0.0780	0.6145	0.0493	0.9264	0.3201
EasyOCR	Sinhala	-	-	-	-	-
	Tamil	0.1172	0.2876	0.3461	0.8828	0.6744

Table 1: The evaluation of OCR systems for the Sinhala and Tamil languages

specifically fine-tuned for Sinhala, while EasyOCR lacks Sinhala support. When evaluating the overall results for both languages, the Cloud Vision API and Document AI produced similar results, but Document AI outperformed the other engines for Tamil across all metrics.

Overall, the best WER results for Tamil are notably higher than those for Sinhala. This observation highlights the disparity between character and word identification. Document AI, despite achieving a very low CER, achieves a comparatively higher WER, indicating that while the system effectively identifies characters, it struggles with word formation and spacing of the Tamil language. This issue is common across all engines and applies to both languages, as systems tend to perform better at recognising characters but struggle with forming words and managing spacing. In contrast, the METEOR and ANLS scores for both languages are relatively high, suggesting a strong alignment in terms of content, word order, and semantic meaning. However, the BLEU scores for Tamil are markedly lower than those of other metrics, likely due to the elevated WER, which results in fewer successful n-gram overlaps.

Performance of Surya on the Sinhala language has been nothing short of extraordinary, emerging as the standout among the others. The metrics clearly illustrate this success, as highlighted in Table 1. When we compare the best WER for Tamil at 11.98% with that of Sinhala at an impressive 2.61% as depicted in Figure 4, the superiority of the accuracy of the Surya engine for Sinhala becomes strikingly apparent. Furthermore, the METEOR and ANLS scores of 0.9723 and 0.9920, respectively, underscore its near-perfect word-level performance.



(a) Sinhala (b) Tamil

Figure 4: WER results for Sinhala and Tamil

The comparison between Subasa OCR and Tesseract is particularly compelling, as Subasa OCR represents a fine-tuned adaptation of the Tesseract 4.0 engine specifically for Sinhala. The authors of Subasa assert that their modifications yield significantly superior results compared to the standard Tesseract 4.0 (Anuradha et al., 2021). The metric evaluations reveal that Tesseract 5.5.0 now outperforms Subasa OCR across all metrics. This indicates that the latest version of Tesseract by Google has made substantial enhancements for the Sinhala language, even in its vanilla form. However, Tesseract's performance in Tamil is not competitive compared to other systems.

As discussed earlier, due to its lack of support for Sinhala, EasyOCR is evaluated solely on Tamil, on which it demonstrated superior performance among the open-source systems we compared. While the results show a notable decline compared to two commercial engines, the contrast with other open-source solutions is significant.

Furthermore, we performed a character-level error analysis for the best models of each language. This analysis involved counting the number of errors per character by comparing the generated text to a reference. In Sinhala, we identified erroneous characters based on a threshold of more than 5,000 errors. Diacritics such as ‘ ◌ා ’, ‘ ◌ු ’, ‘ ◌ි ’, and ‘ ◌් ’ were particularly difficult to identify. In addition, letters such as ‘ න ’, ‘ ක ’, ‘ ය ’, ‘ ර ’, and ‘ ව ’ posed challenges for Surya in terms of accurate detection.

For Tamil, we set a threshold of more than 1,600 errors to identify erroneous characters. The diacritic ‘ ◌் ’ emerged as the most error-prone character, while letters such as ‘ க ’, ‘ த ’, ‘ ள ’, and ‘ ர ’ were also among the most problematic for Document AI.

The number of character errors is generally higher for Sinhala compared to Tamil. However, the best CER scores for both languages do not highlight this difference. This discrepancy arises because, despite having a greater number of character-level errors, the overall usage frequency per character in Sinhala is also comparably higher, which reduces the average error rate. This phenomenon is potentially due to the differences in the sizes of their alphabets[31]: Sinhala has 60 characters, while Tamil contains 247.

5 Conclusion

In this study, we evaluated six different OCR engines for two distinct South Asian languages in a zero-shot setting. To facilitate this evaluation, we created a synthetic Tamil OCR dataset, utilising six different fonts to be parallel to the existing Sinhala dataset. The performance of selected OCR systems was thoroughly analysed using five measurements that evaluated accuracy levels at both the character and word levels.

The results indicated that Document AI performed best for Tamil, while Surya excelled in Sinhala. Both the Cloud Vision API and Document AI showed reasonable performance in OCR for these languages, highlighting the capabilities of commercial engines, as anticipated. A standout performer was Surya for Sinhala, which outperformed all other OCR systems in each metric. Furthermore, the significant disparity between the best CER and WER results for Tamil compared to Sinhala indicates that while Document AI excels at character recognition, it falls short in accurately identifying words through proper character formation and white-space detection. Additionally, zero-shot Tesseract 5.5.0 outperformed a fine-tuned Tesseract 4.0 system on Sinhala (Subasa OCR). Moreover, The differences in scores between the commercial OCR systems are largely a black box, likely arising from nuances in their architectures and training data.

Out of approximately 2.2 million Tamil text records, we selected only 7,000 records to ensure a fair comparison with the Sinhala dataset, which has only 6,969 records. In future studies, it would be possible to expand the two synthetic datasets and consider more fonts, backgrounds, and varied noise conditions to create more realistic simulations.

Limitations

The analysis assessed various OCR engines for Tamil and Sinhala printed scripts without fine-tuning, focusing solely on existing systems. A limitation is that both datasets used were synthetically created, featuring black text on a white background, which does not reflect real-world conditions. Synthetic data was chosen due to the lack of similar realistic datasets for these two low-resourced languages to ensure a fair comparison.

While a considerable amount of work exists for Indic languages such as Hindi, according to the observations of de Silva (2025), the Devanagari script of Hindi has a distance of 5 from Sinhala and a distance of 7 from Tamil, as opposed to a distance of 4 between Sinhala and Tamil. Further, according to Rao (2021), scripts such as Sinhala and Tamil are considered rounded scripts, while Hindi is not. For these reasons, our work could not rely much on the progress made for Hindi OCR.

OCR technology struggles with real-world data due to poor image quality, especially in historical documents and low-resource languages. Factors like low resolution, shading, blurriness, and distortion can severely impact accuracy (Hegghammer, 2022). Issues such as textured backgrounds, cluttered environments, disconnected line segments, isolated dots, breaks in lines, rotation, and motion blur complicate character recognition, leading to higher error rates. Additionally, low resolutions can slow down OCR speed due to uncertainty in character representation (Anuradha et al., 2021). Therefore, the accuracy for camera-captured images may differ notably from this study's results.

[31]The moniker *Alphabet* is used in the general meaning here. Both these scripts are in fact *Abugidas* rather than *Alphabets*.

References

Milind Agarwal and Antonios Anastasopoulos. 2024. A concise survey of OCR for low-resource languages. In *Proceedings of the 4th Workshop on Natural Language Processing for Indigenous Languages of the Americas (AmericasNLP 2024)*, pages 88–102, Mexico City, Mexico. Association for Computational Linguistics.

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. 2023. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4895–4901, Singapore. Association for Computational Linguistics.

Isuri Anuradha, Chamila Liyanage, and Ruvan Weerasinghe. 2021. Estimating the Effects of Text Genre, Image Resolution and Algorithmic Complexity needed for Sinhala Optical Character Recognition. *International Journal on Advances in ICT for Emerging Regions (ICTer)*, 14(3).

Isuri Anuradha, Chamila Liyanage, Harsha Wijayawardhana, and Ruvan Weerasinghe. 2020. Deep Learning Based Sinhala Optical Character Recognition (OCR). In *2020 20th International Conference on Advances in ICT for Emerging Regions (ICTer)*, pages 298–299. IEEE.

Ninad Awalgaonkar, Prashant Bartakke, and Ravindra Chaugule. 2021. Automatic License Plate Recognition System Using SSD. In *2021 international symposium of Asian control association on intelligent robotics and industrial automation (IRIA)*, pages 394–399. IEEE.

Youngmin Baek, Bado Lee, Dongyoon Han, Sangdoo Yun, and Hwalsuk Lee. 2019. Character region awareness for text detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9365–9374.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Ali Furkan Biten, Ruben Tito, Andres Mafla, Lluis Gomez, Marçal Rusinol, Ernest Valveny, CV Jawahar, and Dimosthenis Karatzas. 2019. Scene Text Visual Question Answering. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4291–4301.

P E E Fernando. 1949. Palaeographical Development of the Brahmi Script in Ceylon from 3rd Century BC to 7th Century AD. *University of Ceylon Review*, 7(4):282–301.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Thomas Hegghammer. 2022. OCR with Tesseract, Amazon Textract, and Google Document AI: a benchmarking experiment. *Journal of Computational Social Science*, 5(1):861–882.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Pooja Jain, Kavita Taneja, and Harmunish Taneja. 2021. Which OCR toolset is good and why: A comparative study. *Kuwait Journal of Science*, 48(2).

Guillaume Jaume, Hazim Kemal Ekenel, and Jean-Philippe Thiran. 2019. Funsd: A dataset for form understanding in noisy scanned documents. In *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*, volume 2, pages 1–6. IEEE.

Adithya Kolavi, Samarth P, and Vyoman Jain. 2025. Nayana OCR: A scalable framework for document OCR in low-resource languages. In *Proceedings of the 1st Workshop on Language Models for Underserved Communities (LM4UC 2025)*, pages 86–103, Albuquerque, New Mexico. Association for Computational Linguistics.

Pierre Lison and Jörg Tiedemann. 2016. OpenSubtitles2016: Extracting large parallel corpora from movie and TV subtitles. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 923–929, Portorož, Slovenia. European Language Resources Association (ELRA).

Chamila Liyanage, Thilini Nadungodage, and Ruvan Weerasinghe. 2015. Developing a commercial grade Tamil OCR for recognizing font and size independent text. In *2015 Fifteenth International Conference on Advances in ICT for Emerging Regions (ICTer)*, pages 130–134. IEEE.

YVANT Maduranga and Shantha Jayalal. 2022. Multi-Style Printed Sinhala Character Recognition and Digitalization Using Artificial Neural Network. In *2022 2nd International Conference on Advanced Research in Computing (ICARC)*, pages 120–124. IEEE.

Rishabh Mittal and Anchal Garg. 2020. Text extraction using OCR: A Systematic Review. In *2020 second international conference on inventive research in computing applications (ICIRCA)*, pages 357–362. IEEE.

Meharuniza Nazeem, Anitha R, Navaneeth S, and Rajeev R. R. 2024. Open-source OCR libraries: A comprehensive study for low resource language. In *Proceedings of the 21st International Conference on Natural Language Processing (ICON)*, pages 416–421, AU-KBC Research Centre, Chennai, India. NLP Association of India (NLPAI).

R Paneerselvam. 1972. A critical study of the Tamil Brahmi insciptions. *Acta Orientalia*, 34:35–35.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Surangika Ranathunga, Nisansa De Silva, Dilith Jayakody, and Aloka Fernando. 2024. Shoulders of giants: A look at the degree and utility of openness in NLP research. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 519–529, Bangkok, Thailand. Association for Computational Linguistics.

Surangika Ranathunga and Nisansa de Silva. 2022. Some languages are more equal than others: Probing deeper into the linguistic disparity in the NLP world. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 823–848, Online only. Association for Computational Linguistics.

Kshitij Rao. 2021. How to tell differences between Indian languages (their scripts).

Ransaka Ravihara. 2024. sinhala_synthetic_ocr-large (revision f3cac3b).

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

Nisansa de Silva. 2025. Survey on Publicly Available Sinhala Natural Language Processing Tools and Research. *arXiv preprint arXiv:1906.02358v25*.

Ray Smith. 2007. An overview of the Tesseract OCR engine. In *Ninth international conference on document analysis and recognition (ICDAR 2007)*, volume 2, pages 629–633. IEEE.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in OPUS. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2214–2218, Istanbul, Turkey. European Language Resources Association (ELRA).

Purushoth. Velayuthan and Thanuja D. Ambegoda. 2025. Benchmarking OCR Models for Sinhala and Tamil Document Digitization. Technical report, Engineering Research Unit, University of Moratuwa.

Ruvan Weerasinghe, Asanka Wasala, Dulip Herath, and Viraj Welgama. 2008. NLP applications of Sinhala: TTS & OCR. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-II*.

Haoran Wei, Chenglong Liu, Jinyue Chen, Jia Wang, Lingyu Kong, Yanming Xu, Zheng Ge, Liang Zhao, Jianjian Sun, Yuang Peng, et al. 2024. General OCR Theory: Towards OCR-2.0 via a Unified End-to-end Model. *arXiv preprint arXiv:2409.01704*.

Yudhanjaya Wijeratne, Nisansa de Silva, and Yashothara Shanmugarajah. 2019. Natural Language Processing for Government: Problems and Potential. *International Development Research Centre (Canada)*, 1.