Towards Accent-Aware Vedic Sanskrit Optical Character Recognition Based on Transformer Models

Yuzuki Tsukagoshi
The University of Tokyo / Tokyo, Japan
yuzuki@l.u-tokyo.ac.jp

Ryo Kuroiwa
National Institute of Informatics / SOKENDAI / Tokyo, Japan
kuroiwa@nii.ac.jp

Ikki Ohmukai
The University of Tokyo / Tokyo, Japan
i2k@l.u-tokyo.ac.jp

Abstract

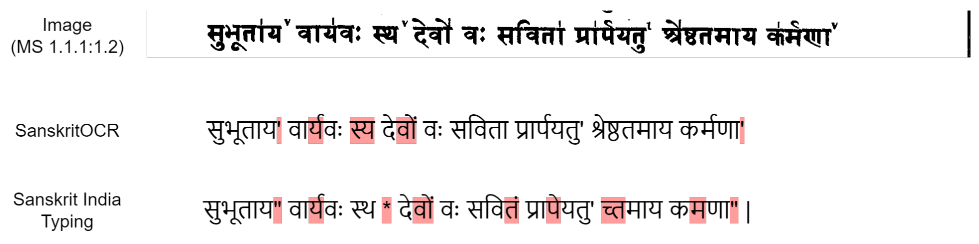
Optical Character Recognition (OCR) for Sanskrit has largely focused on Classical Sanskrit, often neglecting the unique challenges of Vedic Sanskrit, which features essential accent marks for linguistic and philological research. This work proposes an accent-aware OCR system based on Transformer models to recognize printed Vedic Sanskrit texts in Devanagari script and convert them into ISO 15919-compliant Latin script. We leverage state-of-the-art models like TrOCR and a newly curated dataset tailored for Vedic OCR. Aligning OCR outputs with existing electronic texts serves two purposes: facilitating philological research by linking digitized texts with their scanned sources and creating a high-quality, line-based OCR dataset for further improvements. Fine-tuning on domain-specific data significantly enhances accuracy, particularly in recognizing accent marks. Our alignment analysis also suggests high reliability in linking electronic text to scanned images.

1 Introduction

Optical Character Recognition (OCR) systems for Sanskrit often consider the language as a single, unified entity. However, Vedic Sanskrit —which constitutes the language of the Vedas— differs significantly from other variants of Sanskrit, particularly in terms of its rich verbal inflection system and unique accent system. Despite its importance in philological and linguistic research, Vedic Sanskrit has largely been lumped together with Classical Sanskrit in low-resource language studies, and there has been insufficient research focusing specifically on its distinctive accent notation in natural language processing tasks including OCR.

Moreover, Vedic texts remain highly relevant for both philological and linguistic research. Being able to convert existing print editions and as-yet undigitized Vedic manuscripts into accurate machine-readable texts facilitates a range of scholarly endeavors. For instance, linking OCR outputs with existing electronic texts in some text platforms (GRE, ; Martínez García and Gippert, 1995; Kölligan and Reinöhl, 2020; SARIT, 2014; Hellwig, 2010 2021) to build comprehensive digital archives will foster deeper textual comparisons and analyses. Texts with corresponding images are valuable for reading closely Vedic texts, and especially for training OCR models on a diverse range of fonts and styles in books.

In this work, we propose an end-to-end OCR framework to convert printed Vedic Sanskrit texts from Devanagari script with accent marks into ISO 15919-compliant Latin script. Our system recognizes the crucial accent marks that distinguish Vedic Sanskrit from other variants of Sanskrit, making the OCR task more challenging than Classical and other Sanskrit OCR tasks. Beside the OCR system, we also describe the creation of a dataset specifically for Vedic OCR and demonstrate how state-of-the-art Transformer-based approaches, such as TrOCR, can be effectively applied to this unique challenge. Our research is the first that utilizes transformer architecture for Sanskrit OCR tasks. It is simple yet effective for Vedic OCR and easily fine-tunable for Classical Sanskrit OCR and handwritten manuscript OCR tasks.



सुभूताय' वार्यवः स्य देवों वः सविता प्रार्पयतु' श्रेष्ठतमाय कर्मणा'

सुभूताय" वार्यवः स्थ * देवों वः सवितं प्रापेयतु' च्तमाय कमणा" ।

Figure 1: An example of OCR output from a Vedic Sanskrit text.

2 Related Work

Several online OCR services provide Sanskrit character recognition, but their performance is limited. Notable examples include general-purpose web-based OCR systems and specialized websites such as Sanskrit OCR (Sanskrit OCR, 2025) and Sanskrit India Typing (Sanskrit India Typing, 2025), which frequently misrecognize accent marks and confuse character forms like r and the anusvāra (1). Such web-based OCR tools are fundamentally difficult in recognizing large-scale texts and in developing the models further for specialized tasks like Vedic Sanskrit OCR.

Early attempts at Sanskrit OCR primarily relied on rule-based or statistical techniques, which struggled with the complexity of the Devanagari script and its numerous ligatures. With advances in deep learning, recent studies have employed Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) to address these challenges more effectively. However, these approaches often focus on classical Sanskrit and do not consistently account for the unique features of Vedic Sanskrit, such as accent marks.

One prominent example is the line-based OCR approach described by Dwivedi et al. (2020). Their system addresses the challenge of arbitrarily long words, a common phenomenon in Sanskrit due to extensive nominal and verbal compounding. By processing entire lines instead of attempting to segment words as individual units, they reduce segmentation errors that can arise from complex ligatures or sandhi (phonetic transformations). Their model consists of a CNN for image feature extraction and a Bidirectional Long Short-Term Memory (BiLSTM) network for sequence modeling.

A key aspect of their work is the use of synthetic data for dataset augmentation. Sanskrit texts can vary significantly in font style, size, and printing quality; generating artificial images with different fonts and noise levels helps increase robustness and improve generalization. While their results show promising accuracy for Classical Sanskrit documents, the system does not specifically address Vedic documents.

Similarly, Avadesh and Goyal (2018) propose a CNN-BLSTM pipeline for the OCR of Sanskrit manuscripts. Although their primary focus is on historical documents, they employ a comparable architecture – CNN for feature extraction, BiLSTM for sequence decoding– and report high character-level accuracy on Classical Sanskrit samples. As with Dwivedi et al. (2020), their work is indicative of the utility of CNN-BiLSTM methods for Devanagari-based texts but does not fully explore the specialized requirements of Vedic texts with accent diacritics.

In addition to research on OCR mechanisms, post-editing OCR outputs is an active area of study. Sanskrit OCR pipelines often incorporate post-processing steps to correct residual errors, especially in cases where training data is scarce or the input images are noisy. Krishna et al. (2018) introduce a post-editing framework that corrects existing OCR outputs for romanized Sanskrit (in IAST). Their approach leverages CopyNet, a sequence-to-sequence model that learns to transform OCR-generated textinto a corrected output, while Byte-Pair Encoding (BPE) is used to tackle out-of-vocabulary tokens stemming from rich morphological variations in Sanskrit. This work underscores the importance of post-OCR correction, but it is primarily aimed at Classical Sanskrit transliterated into IAST – where accent marks typically used in Vedic contexts

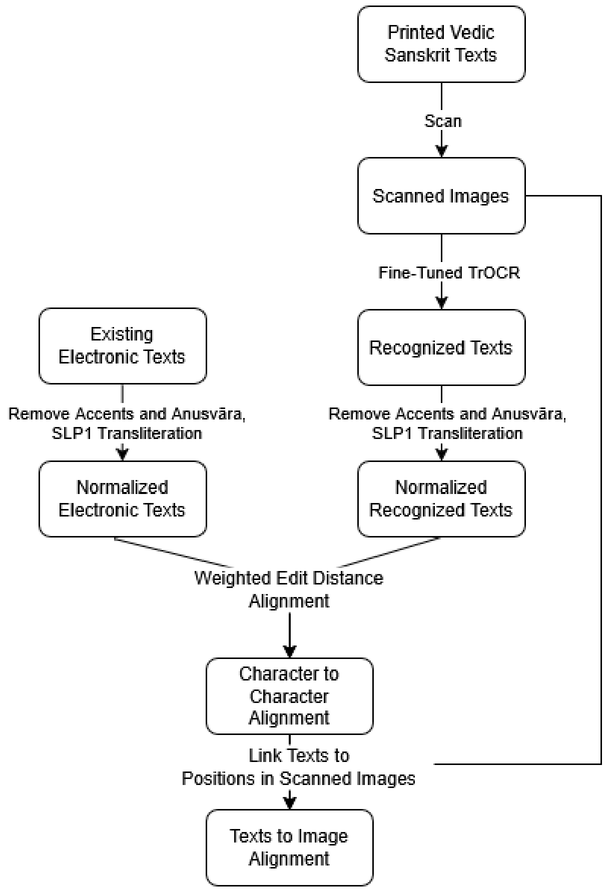

Printed Vedic Sanskrit Texts
Scan
Scanned Images
Fine-Tuned TrOCR
Existing Electronic Texts
Recognized Texts
Remove Accents and Anusvāra, SLP1 Transliteration
Remove Accents and Anusvāra, SLP1 Transliteration
Normalized Electronic Texts
Normalized Recognized Texts
Weighted Edit Distance Alignment
Character to Character Alignment
Link Texts to Positions in Scanned Images
Texts to Image Alignment

Figure 2: Pipeline of our method.

are absent.

Maheshwari et al. (2022) propose a ByT5-based system for post-OCR correction in Sanskrit, utilizing phoneme-based transliteration SLP1. Their dataset and benchmarks facilitate a deeper exploration of error patterns and correction strategies for Sanskrit text, but again, their resources do not include Vedic literatures.

3 Method

We show an overview of our method in Figure 2. We convert scand images of printed Vedic Sanskirt texts into texts by using TrOCR (Li et al., 2023) fine-tuned with Vedic Sanskirt texts. We also align existing electronic texts to the recognized texts based on weighted edit distance (Das et al., 2023) after normalizing them by removing accents and anusvāra and SLP1 transliteration. Based on the text alignment, we link the electronic texts to page/line positions in the scanned images. In what follows, we provide a detailed explanation of each step.

3.1 Dataset

For our experiments, we use a printed edition of the Maitrāyaṇī Saṁhitā (MS), edited by Schroeder (Schroeder, 1881), and an electronic text input by Fushimi[1]. The digital edition contains page and line numbers in the originial printed edition, making itself suitable for line-based OCR, though the motivation to annotate page and line numbers is not for OCR but for philological purposes. Although the long-term goal is to handle various fonts and shapes of characters in printed books and handwritten manuscripts, we begin with printed text to simplify the initial OCR task.

[1]https://titus.uni-frankfurt.de/texte/etcs/ind/aind/ved/yvs/ms/ms.htm

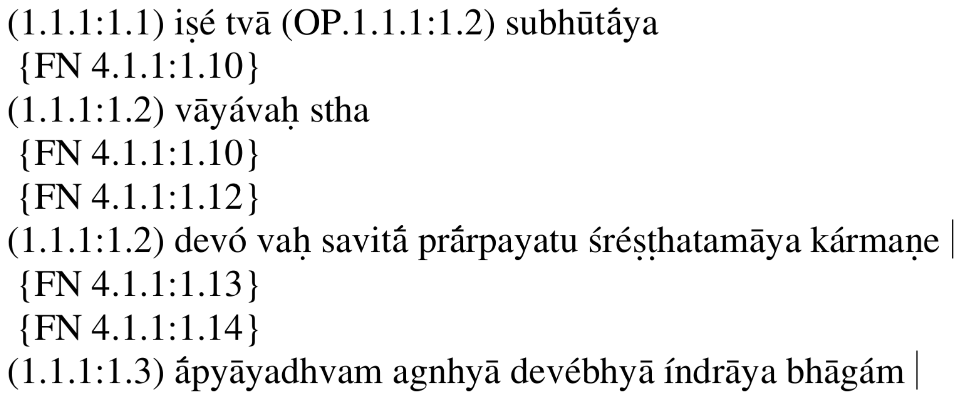

(1.1.1:1.1) iṣé tvā (OP.1.1.1:1.2) subhūtā́ya
{FN 4.1.1:1.10}
(1.1.1:1.2) vāyávaḥ stha
{FN 4.1.1:1.10}
{FN 4.1.1:1.12}
(1.1.1:1.2) devó vaḥ savitā́ prā́rpayatu śrḗṣṭhatamāya kármaṇe |
{FN 4.1.1:1.13}
{FN 4.1.1:1.14}
(1.1.1:1.3) ā́pyāyadhvam agnhyā devébhyā índrāya bhāgám |

Figure 3: An example of the Fushimi's electronic text of the MS.

The printed edition of MS is in Devanagari script with accent marks, which apear as vertical strokes above the character. The edition consists of four books, each with more than 200 pages summing up to about 900 pages with preface and indices.

The first step to preparing the dataset is to convert the PDF file into page images. We then split the page images into line images using line segmentation method based on haar-like features (Viola and Jones, 2001). After splitting into line images, we manuaaly corrected the line images to remove errors in line segmentation. We finally got 11,740 correctly segmented line images from the 900 pages of the MS.

Fushimi's electronic text has page and line numbers with MS text numbers in the initial part of each line (3). The part after the colon in the parentheses denotes the page and line numbers in the MS: 1.1 means the first page and the first line of the Schoroeder's edition, 1.2 means the first page and the second line, and so on. The text has also a significant feature that words are separeted by spaces. When a word ends with a consonant and the next word starts with a vowel, the orignial Devanagari text has one character for the consonant and the vowel. Although the way of separating words reflects Fushimi's philological understanding of the text, we adopt the separation for the end-to-end OCR system. Accounting for the structure of the electronic text, we prepared textual material with a simple script to extract page and line numbers and the text from the electronic text.

3.2 Transliteration and Accent Marks

Our OCR model is designed to recognize Devanagari text into Latin script following ISO 15919. while IAST, a conventional transliteration method, remains common. The diffrence between these two transliteration schemes is that ISO 15919 uses 'r̥' for vocalic 'r' and 'l̥' for vocalic 'l', whereas IAST uses 'ṛ' for vocalic 'r' and 'ḷ' for vocalic 'l'. The reason we chose ISO 15919 is to accommodate certain Vedic phonemes (e.g., the Vedic retroflex l transliterated as 'ḷ' distinct from 'ḍ') that are not adequately handled by IAST in which vocalic 'l' is transliterated as 'ḷ'. Furthermore, ISO 15919 is more suitable for cross-linguistic OCR tasks, as it is a standard for Indic scripts.

Retaining and accurately recognizing Vedic accent marks (e.g., svarita, udātta) is paramount, as these accentual distinctions are critical for proper linguistic and philological analysis of the Vedas.

3.3 Architecture: TrOCR

We adopt TrOCR (Li et al., 2023) that is based on the state-of-the-art Transformer architecture (Vaswani et al., 2017). It is fully a transformer model consisting of a visual transformer encoder and a text transformer decoder. This approach has demonstrated high performance in a wide range of OCR tasks. In the environment where we fine-tune transformer models with

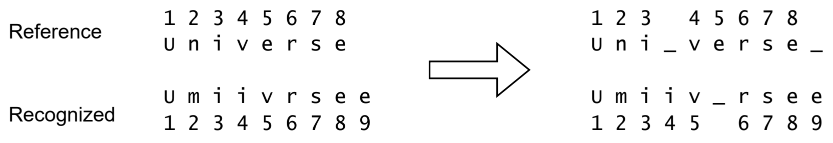


Figure 4: An example of an alignment.

ease, TrOCR is a suitable choice for our Vedic OCR task. For the difficulty of our task on Vedic character recognition, we use a large-sized pre-trained model for printed text recognition microsoft/trocr-large-printed [2], and fine-tune it on our Vedic OCR dataset. The fine-tuned model is publicly available[3].

Our experiments were conducted on two NVIDIA GPU (RTX A6000 with 48GB memory). We fine-tuned the model using the following hyperparameters:

- Max Sequence Length: 512 tokens
- Train Batch Size: 16
- Eval Batch Size: 16
- Learning Rate: 2×10^{-5}
- Weight Decay: 0.01
- Number of Training Epochs: 20

The sequence length of 512 tokens is sufficient to capture most lines of text without truncation. We chose a train batch size and eval batch size of 16 based on available GPU memory. The learning rate of 2×10^{-5} and weight decay of 0.01 follow standard recommendations for Transformer-based OCR models. We ran fine-tuning for 20 epochs, noting that early stopping or further tuning might be beneficial depending on the dataset size and distribution.

3.4 Post-OCR Matching and Alignment

After running the OCR system, we align the recognized lines with existing electronic texts for error correction and verification. As an illustrative example, we use the alignment of the first book of Maitrāyaṇī Saṁhitā.

3.4.1 Weighted Edit Distance Alignment

We represent the texts as sequences of characters, $X = (x_1, ..., x_n)$ and $Y = (y_1, ..., y_m)$. In an alignment, the i-th position of X is mapped to the j-th position of Y or not mapped to any position. No two positions in X are mapped to the same position in Y, but it is possible that multiple positions in X are not mapped to any positions. In addition, not all positions in Y are used in the alignment. In our case, the existing electronic texts correspond to X, and the recognized texts correspond to Y. If the i-th position of X is not mapped to any position, then that character is considered missing in the recognized texts. If no position in X is mapped to the j-th position of Y, the j-th character of Y is considered to be mistakenly inserted. We present an example of alignment in Figure 4.

We consider an alignment to be the transformation of a sequence $X = (x_1, ..., x_n)$ into another sequence $Y = (y_1, ..., y_m)$ by substituting, deleting, and inserting characters in X. To find a

[2]https://huggingface.co/microsoft/trocr-large-printed
[3]https://huggingface.co/yzk/trocr-large-printed-vedic

good alignment, we consider finding one that minimizes the weighted edit distance (Das et al., 2023), which requires cost $w(x_i, y_j)$ to substitute a character x_i with y_j, cost $w_{\mathrm{del}}(x_i)$ to delete a character x_i, and cost $w_{\mathrm{ins}}(y_j)$ to insert a character y_j. Intuitively, the weight $w(x_i, y_j)$ is small if two characters x_i and y_j are similar, $w_{\mathrm{del}}(x_i)$ is small if x_i is likely to be lost by OCR, and $w_{\mathrm{ins}}(y_j)$ is small if y_j is likely to be inserted.

Let $d(i, j)$ be the weighted edit distance between $(x_1, ..., x_i)$ and $(y_1, ..., y_j)$, i.e., considering up to the i-th and j-th characters in X and Y, respectively. Intuitively, to align $(x_1, ..., x_i)$ and $(y_1, ..., y_j)$, we consider three possibilities:

1. Align $(x_1, ..., x_{i-1})$ with $(y_1, ..., y_j)$ and delete x_i.
2. Align $(x_1, ..., x_i)$ with $(y_1, ..., y_{j-1})$ and append y_j.
3. Align $(x_1, ..., x_{i-1})$ with $(y_1, ..., y_{j-1})$ and map x_i to y_j.

More formally, $d(i, j)$ can be computed by the following recursive equation:

$$d(i,j) = \min \begin{cases} d(i-1,j) + w_{\mathrm{del}}(x_i) & i > 0 \\ d(i,j-1) + w_{\mathrm{ins}}(y_j) & j > 0 \\ d(i-1,j-1) + w(x_i,y_j) & i > 0, j > 0 \end{cases} \tag{1}$$

where $d(0, 0) = 0$. We assume that $w(x_i, y_j) = 0$ if $x_i = y_j$. The weighted edit distance between X and Y corresponds to $d(n, m)$.

With Equation (1), the weighted edit distance and the corresponding alignment can be computed by dynamic programming (Bellman, 1957). A naive approach is to compute $d(i, j)$ for $j = 0, ..., m$ starting from $i = 0$ and then compute $d(i + 1, j)$ using $d(i, j)$, $d(i, j - 1)$, and $d(i + 1, j - 1)$. In practice, we use a more sophisticated dynamic programming algorithm proposed by previous work (Ukkonen, 1985), assuming that $w_{\mathrm{del}}(x_i) \geq 1$, $w_{\mathrm{ins}}(y_j) \geq 1$, and $w(x_i, y_j) \geq 0$ for any x_i and y_j. With such an assumption, $d(i, j)$ is lower bounded by $|i - j|$, and if the weighted edit distance is less than or equal to k, we can compute it by enumerating $d(i, j)$ with $|i - j| \leq k$. Starting from $k = |n - m| + 1$, we try to compute the weighted edit distance by such enumeration and double k if we fail.

For constructing an alignment, for each (i, j), we record which operation is used to compute $d(i, j)$, i.e., deletion if $d(i, j) = d(i - 1, j) + w_{\mathrm{del}}(x_i)$, insertion if $d(i, j) = d(i, j - 1) = w_{\mathrm{ins}}(y_j)$, and substitution if $d(i, j) = d(i - 1, j - 1) + w(x_i, y_j)$. When different operations result in the same weighted edit distance, we prioritize deletion over the other two and insertion over substitution; assuming the transformation from the existing electronic texts into the recognized texts by OCR, our intuition is that the deletion of characters is likely to happen by skipping lines or pages, and the substitution is also possible, but insertion is less frequent.

3.4.2 Text Normalization

We normalized the reference and OCR-recognized texts for an alignment. The recognized text contains errors due to the difficulty of the recognitions of accent marks, and anusvāra displayed as a dot in the image.

For the purpose of making an alignment easy, we conducted the following three normalization process. First, because the error rate of the recognition of accent marks is considered high, we removed the accent marks in the romanized texts. Similarly the second process is to remove anusvāra that is difficult to recognize from the images. Third, we used SLP1 transliteration method, a way to represent a phoneme as an ASCII character. As a result we removed udātta represented by acute accent in the romanized texts, svarita by grave, and anusvāra.

3.4.3 Weight Parameters

For substitution, we use $w(a, b) = 0$ if $a = b$. In addition, we use zero weights for pairs of similar characters, which are shown in Table 1. Each row presents a set of similar characters when they are written in Devanagari, both in representations of ISO 15919 and SLP1. For any two pairs of characters (a, b) in each line, we use $w(a, b) = w(b, a) = 0$. Otherwise, we use $w(a, b) = 1$ for different characters a and b. For deletion and insertion, we use a unit cost, i.e., $w_{\text{del}}(a) = 1$ and $w_{\text{ins}}(a) = 1$ for any a.

ISO 15919	SLP1
gh, dh	G, D
c, b, v	c, b, v
p, bh, m, y, s, ṣ	p, B, m, y, s, S

Table 1: A list of characters whose weights are zero.

4 Results

4.1 OCR result

Table 2 presents our primary evaluation metrics before and after fine-tuning (FT) the microsoft/trocr-large-printed model on the Maitrāyaṇī Saṁhitā (MS) dataset. We report the Character Error Rate (CER), Word Error Rate (WER), and the CHRF1 score (with parameters character order = 6, word order = 0, beta = 2).

Metric	Scope	Before FT	After FT
Character Error Rate (CER)	Full	0.9872	0.0737
	Base	0.9870	0.0690
Word Error Rate (WER)	Full	1.0000	0.2916
	Base	1.0000	0.2787
CHRF1 Score	Full	0.3893	85.38
	Base	0.4190	86.71

Table 2: Comparison of model performance metrics before and after fine-tuning (FT) on the MS data (CHRF1 parameters are character order = 6, word order = 0, beta = 2.). "Full" denotes metrics computed considering accents; "Base" denotes metrics computed after removing acute and grave accents.

Prior to fine-tuning (Before FT), the model demonstrated extremely high error rates, both at the character level (CER = 0.9872) and the word level (WER = 1.0000). This indicates that the off-the-shelf TrOCR model, trained mostly on modern printed texts, fails to generalize to Sanskrit Devanagari. The near-zero CHRF1 score (0.3893) further reflects the severe mismatch between the model's predictions and the ground truth.

By contrast, once the model was fine-tuned (After FT) on line-level data from the MS edition, the CER dropped to 0.0737 and the WER fell to 0.2916. These improvements are also mirrored in the CHRF1 score, which soared to 85.38. In contrast to the accent-aware evaluation (Full), the base letter evaluation (Base) yielded even lower error rates. That implies that it is difficult to recognize accent marks in the Vedic Sanskrit texts.

Additionally, we evaluated the model's performance specifically on recognizing the positions of accent marks. The results are summarized in Table 3. Accent mark recognition was evaluated on a syllable-by-syllable basis, where only vowels capable of carrying accents were considered. Overall, the results illustrate that domain-specific training for Vedic Sanskrit – where accent marks play a crucial role – dramatically boosts accuracy.

The findings imply that the TrOCR model, pre-trained on a large amount of printed text, cannot effectively recognize Devanagari, which is dominantly used in the modern Indian subcontinent. The necessity of fine-tuning from basical Devanagari documents is highlighted. This underscores the importance of assembling appropriate datasets that capture script intricacies, varied fonts, and unique diacritical elements for each domain.

	Before FT	After FT
Precision	0.0000	0.7110
Recall	0.0000	0.6663
F1	0.0000	0.6879

Table 3: Metrics for accent mark recognition.

4.2 Alignment result

Since we use the weight edit distance alignment, each position of a character in the existing electronic texts is mapped to a position in the recognized texts or is not mapped to any position. However, in actual use case, it is sufficient to get a page and a line number in which an initial part of the target sentence is contained. Then, Vedic experts can find the image corresponding to the digitized texts.

With this motivation, from an alignment, we extract the page and the line to which the initial part of a sentence is aligned. Given a sentence starting from position i in the electronic texts, we get the page and the line containing the position j in the recognized texts to which i is aligned. If i is not aligned to any position, then we find the first position i' that is aligned to a position j'. If i' is in the same sentence as i, we get the page and the line containing j'. Otherwise, we consider that the sentence is missing in the recognized texts due to scanning errors or OCR layout errors.

Table 4 presents the alignment statistics on the test data. The electronic text contains a total of 5,844 sentences. The number of sentences whose starting line appears in the OCR-recognized text is 5094. The number of correctly aligned sentences is 5048, while 35 were misaligned to an incorrect line. Additionally, 11 sentences were mistakenly identified as missing. There are 750 sentences that is missing due to the scan or layout detection errors. In our definition, the correct identification of missing images is to indicate that a sentence has no corresponding match. Based on this, 271 sentences were correctly marked as missing, while 479 sentences failed to be identified as missing.

	Total	#correctly aligned	#wrongly aligned	#not aligned
Sentences in the electronic text	5844	5048	514	282
Included in the OCR text	5094	5048	35	11
Lost in the OCR text	750	-	479	271

Table 4: Breakdown of the alignment operations

The accuracy was calculated as the number of correctly identified sentences relative to the total number of sentences. To evaluate the accuracy of sentence alignment, we calculated the proportion of sentences without missing images that were correctly aligned, and the accuracy was $5094/5844 = 0.8637$. And the accuracy, including sentences with missing images, is $(5048 + 282)/5844 = 0.9101$.

5 Discussion

Qualitative analysis of the errors reveals common confusions among visually similar Devanagari characters, such as gh and dh, and an especially high error rate for accent symbols (5). Furthermore, we observe a curious phenomenon in which the original image occasionally lacks an accent mark even though linguistic knowledge suggests it should be present, and yet the model infers and inserts an accent in the output. Figure 6 illustrates this scenario: In some cases, our model predicts words without accents as the original image lacks them because of print quality, but naturally accents should be present in the words. In other cases, even though the scanned text shows no accent marks for a given character (e.g., sárvā, tā̃), the OCR output includes an accent based on the model's learned linguistic context. The ability to infer missing accents is a promising feature of the model, even if the performance is not perfect.

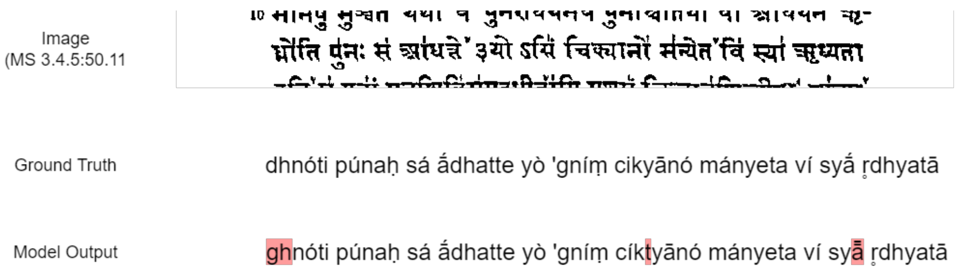


Image (MS 3.4.5:50.11

Ground Truth: dhnóti púnaḥ sá ā́dhatte yò 'gním cikyānó mányeta ví syā́ ṛdhyatā

Model Output: ghnóti púnaḥ sá ā́dhatte yò 'gníṃ cíktyānó mányeta ví syā̄ ṛdhyatā

Figure 5: An example of OCR errors in recognizing dh and gh.

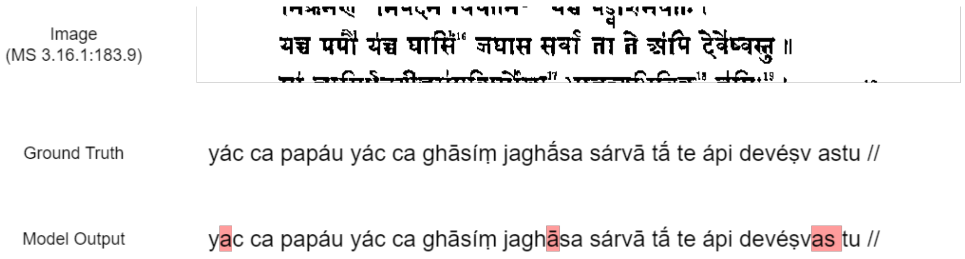


Image (MS 3.16.1:183.9)

Ground Truth: yác ca papáu yác ca ghāsíṃ jaghā́sa sárvā tā́ te ápi devéṣv astu //

Model Output: yac ca papáu yác ca ghāsíṃ jaghāsa sárvā tā́ te ápi devéṣvas tu //

Figure 6: An example of OCR errors in recognizing accent marks.

These errors and a little lack of reasoning ability stem in part from limited training data, indicating that expanding and diversifying the dataset remains a critical next step. While fine-tuning has significantly reduced the overall error rates, these particular challenges suggest that more careful curation and augmentation of Vedic-specific training examples are essential. In particular, accent marks often appear close to other characters, making them more prone to misidentification when the print is faint or if the font style is crowded.

Moreover, some of these difficulties may be attributed to subtle differences in typography between pages or even within the same printed edition. Slight variations in ink thickness or spacing can confuse the model, especially in conjunction with the complex ligature forms common in Devanagari script. Additional data –spanning multiple editions or fonts– could help the model better generalize to such typographic variability.

In our analysis of sentence alignment, we examined cases where OCR misidentified or failed to recognize certain pages and lines due to image loss. There is 11 instances where the sentence was actually present though the judge indicates that the sentence is missing. This suggests that in approximately $11/(11 + 271) \simeq 0.04(4\%)$ of cases, our alignment system incorrectly reports missing content. Conversely, when a sentence is misaligned and led to an incorrect location, the probability that the sentence is truly missing in the original image is $479/(479 + 35) \simeq 0.93(93\%)$. This indicates that when a misalignment occurs, it is highly likely that the sentence is already absent at the OCR stage.

From these results, we can infer that when our alignment system reports a sentence as missing, it is reliable in approximately 96% of cases. Additionally, when a sentence is incorrectly aligned, there is a 93% likelihood that the issue originates from OCR failing to recognize the pages due to prior image loss. These accuracy suggests that when attempting to identify the corresponding image part from the electronic text, the accuracy is useful in practice.

Several directions for improvement remain. First, expanding the volume and variety of training data –especially from different printed editions and fonts– will help address lingering issues of font dependency and accent misrecognition. Second, adopting data augmentation strategies, such as synthetically generating line images with controlled noise and varying accent positions, may further bolster the model's ability to handle out-of-distribution examples. Third, more sophisticated post-OCR correction methods could be integrated, particularly those tailored to handle the nuanced diacritical system of Vedic Sanskrit. Taken together, these improvements will help produce robust digitized texts that preserve the essential accent information required for accurate philological and linguistic analyses of Vedic literature.

6 Conclusion

Vedic Sanskrit OCR is essential for advancing textual and linguistic research in this historically and philologically significant tradition. By focusing on accent-aware recognition and employing a robust transliteration scheme such as ISO 15919, we have demonstrated that Transformer-based approach can significantly improve the readability and accuracy of digitized Vedic texts. Our fine-tuning experiments underscore the importance of domain-specific training data — particularly for recognizing the intricate accent marks that distinguish Vedic Sanskrit from other variants of Sanskrit.

Beyond immediate scholarly applications, the increased availability of digitized Vedic texts can help build more advanced language models, which, in turn, facilitate broader research initiatives in digital humanities and computational linguistics. Although various editions of Vedic literature are already electronically available, many lack post modification or rely on a single printed source. We argue that accurately digitizing additional editions and, ultimately, handwritten manuscripts is vital for producing comprehensive, reliable datasets.

Importantly, our method preserves the full range of Vedic accents, ensuring compatibility with ongoing philological and linguistic studies. Moreover, adopting ISO 15919 opens avenues for cross-linguistic OCR endeavors in the modern Indian languages and the ancient Indo-European languages. At the same time, existing post-editing frameworks cannot be directly applied to Vedic texts, highlighting the pressing need for specialized systems designed to handle the nuanced accentual features of Vedic Sanskrit.

In this light, our work demonstrates the promise of Transformer-based OCR models for under-resourced, accent-rich languages, while advocating for continued development of accent-sensitive approaches in both data preparation and post-processing.

Acknowledgements

This work was supported by JSPS KAKENHI Grant Number 23K18646 and by ROIS NII Open Collaborative Research 2025 (251FP-22775). We sincerely thank Philip Wellnitz and Kaito Fujii for their valuable discussions and insightful comments, which greatly contributed to the improvement of this work.

References

Meduri Avadesh and Navneet Goyal. 2018. Optical character recognition for sanskrit using convolution neural networks. In 2018 13th IAPR International Workshop on Document Analysis Systems (DAS), pages 447–452.

Richard Bellman. 1957. Dynamic Programming. Princeton University Press.

Debarati Das, Jacob Gilbert, MohammadTaghi Hajiaghayi, Tomasz Kociumaka, and Barna Saha. 2023. Weighted edit distance computation: Strings, trees, and dyck. In Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, page 377–390, New York, NY, USA. Association for Computing Machinery.

Agam Dwivedi, Rohit Saluja, and Ravi Kiran Sarvadevabhatla. 2020. An ocr for classical indic documents containing arbitrarily long words. In 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pages 2386–2393.

Göttingen register of electronic texts in indian languages and related indological materials from central and southeast asia.

Oliver Hellwig. 2010–2021. DCS - The Digital Corpus of Sanskrit.

Daniel Kölligan and Uta Reinöhl. 2020. Vedaweb - online research platform for old indic texts.

Amrith Krishna, Bodhisattwa P. Majumder, Rajesh Bhat, and Pawan Goyal. 2018. Upcycle your OCR: Reusing OCRs for post-OCR text correction in Romanised Sanskrit. In Anna Korhonen and Ivan Titov, editors, Proceedings of the 22nd Conference on Computational Natural Language Learning, pages 345–355, Brussels, Belgium, October. Association for Computational Linguistics.

Minghao Li, Tengchao Lv, Jingye Chen, Lei Cui, Yijuan Lu, Dinei Florencio, Cha Zhang, Zhoujun Li, and Furu Wei. 2023. TrOCR: Transformer-Based Optical Character Recognition with Pre-trained Models. Proceedings of the AAAI Conference on Artificial Intelligence, 37(11):13094–13102, June. Number: 11.

Ayush Maheshwari, Nikhil Singh, Amrith Krishna, and Ganesh Ramakrishnan. 2022. A benchmark and dataset for post-OCR text correction in Sanskrit. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, Findings of the Association for Computational Linguistics: EMNLP 2022, pages 6258–6265, Abu Dhabi, United Arab Emirates, December. Association for Computational Linguistics.

Francisco Javier Martínez García and Jost Gippert. 1995. Thesaurus indogermanischer text- und sprachmaterialien.

Sanskrit India Typing. 2025. Sanskrit India Typing. https://sanskrit.indiatyping.com/. Accessed: April 30, 2025.

Sanskrit OCR. 2025. Sanskrit OCR. https://ocr.sanskritdictionary.com/. Accessed: April 30, 2025.

SARIT. 2014. SARIT: Search and Retrieval of Indic Texts. SARIT provides digital tools for philological research into Indic texts.

Leopold von Schroeder. 1881. Maitrāyaṇī Saṃhitā. Die Saṃhitā der Maitrāyaṇīya-Śākhā. Verlag der Deutschen Morgenländischen Gesellschaft, Leipzig.

Esko Ukkonen. 1985. Algorithms for approximate string matching. Information and Control, 64(1):100–118. International Conference on Foundations of Computation Theory.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc.

P. Viola and M. Jones. 2001. Rapid object detection using a boosted cascade of simple features. In Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, volume 1, pages I–I.