An OCR for Classical Indic Documents Containing Arbitrarily Long Words

Agam Dwivedi
dwivedi.agam233@gmail.com

Rohit Saluja
rohitsaluja22@gmail.com

Ravi Kiran Sarvadevabhatla
ravi.kiran@iiit.ac.in

Centre For Visual Information Technology (CVIT)
International Institute of Information Technology, Hyderabad (IIIT-H)
Gachibowli, Hyderabad 500032, INDIA
https://github.com/ihdia/sanskrit-ocr

Abstract

OCR for printed classical Indic documents written in Sanskrit is a challenging research problem. It involves complexities such as image degradation, lack of datasets and long-length words. Due to these challenges, the word accuracy of available OCR systems, both academic and industrial, is not very high for such documents. To address these shortcomings, we develop a Sanskrit specific OCR system. We present an attention-based LSTM model for reading Sanskrit characters in line images. We introduce a dataset of Sanskrit document images annotated at line level. To augment real data and enable high performance for our OCR, we also generate synthetic data via curated font selection and rendering designed to incorporate crucial glyph substitution rules. Consequently, our OCR achieves a word error rate of 15.97% and a character error rate of 3.71% on challenging Indic document texts and outperforms strong baselines. Overall, our contributions set the stage for application of OCRs on large corpora of classic Sanskrit texts containing arbitrarily long and highly conjoined words.

1. Introduction

Optical Character Recognition (OCR) forms an essential component in the workflow of document image analytics. As with other document-related tasks, the advent of deep learning has produced sophisticated and reliable OCR systems for many script systems worldwide [1, 18, 20, 21, 22, 34, 39]. However, barring recent exceptions [1, 4, 14, 24], progress has been less than satisfactory for the numerous scripts from Indian subcontinent.

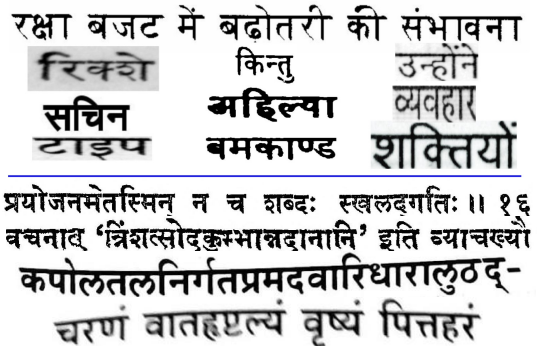

Figure 1: Some images from previous works [16, 19, 23, 28] (above the blue line) and our work (below). Observe the differences in word length and the uneven line alignment in our case.

One reason for the slow progress is the unique challenges with Indic scripts [7, 29]. In many Indic scripts, two or more characters often combine to form conjuncts which considerably increase the vocabulary to be tackled by OCR systems [3, 9]. Furthermore, the visual appearance of conjunct characters is generally more complicated than the individual elementary script characters. Compounding the challenge, the glyph substitution rules for conjunct characters lack consistency across fonts (refer Figure 2).

Even within Indic documents, those written in the classical language Sanskrit exhibit the highest levels of complexity and variety in terms of conjunct characters. Additionally, these documents, typically rendered in Devanagari, routinely contain sentences where words themselves conjoin to arbitrary lengths (see Figure 1). Due to this

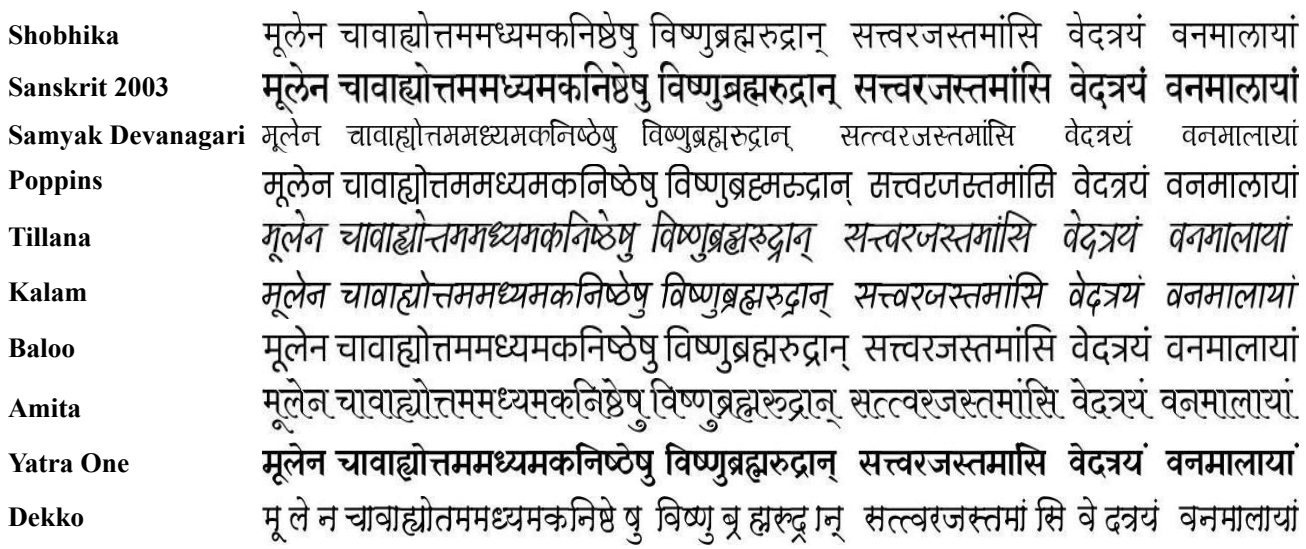

Figure 2: Synthetic images for a Sanskrit sentence in 10 different fonts. Observe that fonts, while seemingly consistent, may contain rendering issues. E.g., note the breaks in shirorekha (the line joining characters in a Devanagari word) in some of the conjunct characters in the Dekko font. Also, in Yatra One font, the dot at the end of last 2 words are slightly misplaced.

phenomenon, approaches relying on character-level processing [2, 11, 12, 15] are not viable. The same is the case with more recent approaches which assume vernacular word-level segmentation and annotation [16, 19, 29]. Apart from academic approaches, open [36] and commercial [10] document-level OCR systems are available for Devanagari. However, as we shall show, these approaches do not work satisfactorily across the range of conjuncts and word-lengths present in Sanskrit documents. Another fundamental issue is the lack of annotated Sanskrit *document image* datasets itself.

To address limitations mentioned above, we propose an OCR for classical Indic documents containing arbitrarily long conjoined words. In our model, a Convolutional Neural Network (CNN) followed by a Bi-directional Long Short Term Memory (BLSTM) encode the input image sequence. Another LSTM then separates the essential parts of the encoded sequence via a single-headed soft-attention mechanism and decodes to obtain the output text (Sec. 4). On the data front, we introduce a Sanskrit document image dataset with multiple classical texts annotated at line level. In addition, we use carefully prepared synthetic data to augment real data and to increase the coverage of our OCR (Sec. 3).

For code, pre-trained models and other resources, visit https://github.com/ihdia/sanskrit-ocr.

2. Related Work

Attempts to recognize Devanagari date to late 70s and were mostly confined to recognition of disjoint characters [30, 35]. The next wave of works utilized larger training corpora , various combinations of hand-crafted feature extraction and classification approaches for improved performance. Bansal *et al.* [2] combine diverse knowledge sources for Devanagari OCR. Jawahar *et al.* [11] use a DAG-SVM [25] to recognize separated character images. Kompalli *et al.* [15] enhance the accuracy of character segmentation based approaches by applying recognition driven segmentation. Sankaran *et al.* [28, 29] further improve text recognition on Hindi word images. Mathew *et al.* [19] show improved results for Devanagari while presenting a multilingual word-level OCR system involving Recurrent Neural Networks (RNNs).

Motivated by the lower annotation burden in obtaining line-level annotations and better context affordance for character recognition, approaches have been proposed for obtaining OCR line-level outputs in documents [4, 14, 24]. Karayil *et. al.* [13] train a line-level segmentation-free 1-d Long Short Term Memory (LSTM) model on vernacular Hindi document images which typically contain shorter words relative to Sanskrit texts. Sanskrit-specific OCRs are still limited to character and conjunct character images [1, 26, 31]. To the best of our knowledge, we propose the first line-level OCR for Sanskrit documents. Our proposed approach operates at line-level using a single-headed attention-based bi-directional LSTM. In addition, our approach can deal with a much larger range of word lengths across larger document corpora.

The use of synthetic data to assist the development of OCR systems is well documented. Karayil *et. al.* [13] train a 1-d Long Short Term Memory (LSTM) model with syn-

Book	Pages	Lines	Words
Nirnaya Sindhu	379	13691	95549
Kavyaprakasha of Mammata	611	6919	26841
Kshemakutuhalam	525	3238	17716
Total	**1515**	**23848**	**140, 106**

Table 1: Statistics of our annotated datasets.

thetic as well as real Hindi lines. This work uses 7 fonts for rendering 1000 synthetic line images and 621 real line images for training 1-D LSTM. Dutta *et. al.* [5] train a CNN-RNN hybrid model for small length *handwritten* word images using synthetic word images rendered with 100 Unicode fonts for pre-training. We use synthetic data as part of our approach as well. However, the procedure requires greater care compared to existing approaches since we need to ensure font-level support for conjunctions.

3. Datasets

We have two sources of data – existing Sanskrit document image texts and synthetically rendered Sanskrit texts.

Document image texts: We annotated 24,000 lines from three different classical Sanskrit texts – Nirnaya Sindhu, Kavyaprakasha of Mammata, Kshemakutuhalam. The annotations were gathered from Sanskrit domain experts to ensure good data quality. The related statistics can be found in Table 1.

Synthetically Rendered texts: We first gathered multiple fonts from various sources [8, 6, 33, 37]. We then shortlisted 67 fonts which enable proper rendering of conjunct characters. To maximize diversity of synthetic data, we sourced 5000 unique lines per font from classical Sanskrit texts found at https://sanskritdocuments.org. We particularly selected lines containing long words and complex conjuncts. By rendering text lines using these fonts, we obtained around $335k$ (67×5000) synthetic line images with associated text ground truth to train our model. Figure 2 depicts synthetic images of a Sanskrit sentence in 10 different fonts from our chosen set.

Figure 3 depicts the word distribution in terms of word length across our synthetic and real documents on a logarithmic scale. As can be seen, the real dataset (in red) covers only a small fraction of long words. Our synthetic dataset (in blue) greatly increases the coverage of words across word lengths. As Figure 3 also shows, some larger word-lengths missing from real data are compensated by presence via synthetic data.

3.0.1 Data Preparation

We use binary images for our synthetic data and crop the rendered text to remove extra whitespace surrounding the

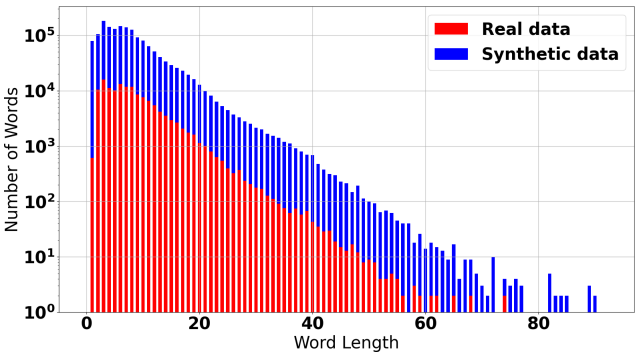

Figure 3: Coverage of words in our dataset. Note that 'Number of words' (y-axis) is logarithmic scale.

actual text content. As mentioned before, we curate the rendered text to ensure that all font synthesized images look similar to the original text with regards to glyph substitutions/conjoined characters. We use `pango`[1] and `cairo`[2] libraries for text-rendering of synthetic images with properly conjoined characters.

The vocabulary set consists of 144 characters (129 Devanagari characters, 10 numeric digits, and 5 punctuation symbols). The images are all normalized to a fixed height of 32 pixels with the width adjusted according to original aspect ratio.

4. Model

Sanskrit texts have the highest fraction of out of vocabulary words among documents in different Indic languages [27]. In the field of photo OCR, lexicon-free recognition has achieved great benefits using attention-based models [17]. Therefore, we use an attention-based model that works well for lexicon-free recognition tasks. Specifically, we use a CNN to extract image features and feed them to a Bi-directional Long Short Term Memory (BLSTM) model to encode the input image. We use another LSTM with single-headed attention mechanism to decode the encoded features.

The input images are padded to a fixed width of 512 and fed to a 7-layer CNN network (see Table 2) to obtain line image features $\{c_1, c_2, ..., c_n\}$. The CNN features are used as input for encoder RNN which is a BLSTM model with 256 hidden units. The decoder is a two-layer LSTM (shown at the top of Figure 4) with each layer having 128 hidden units.

Following [38], the attention mechanism operates on BLSTM's hidden state sequence h ($= \{h^1, h^2, ..., h^n\}$) in conjunction with the hidden state representation from the Decoder LSTM to obtain attention coefficients α_t (shaded

[1] https://pango.gnome.org/
[2] https://cairographics.org/

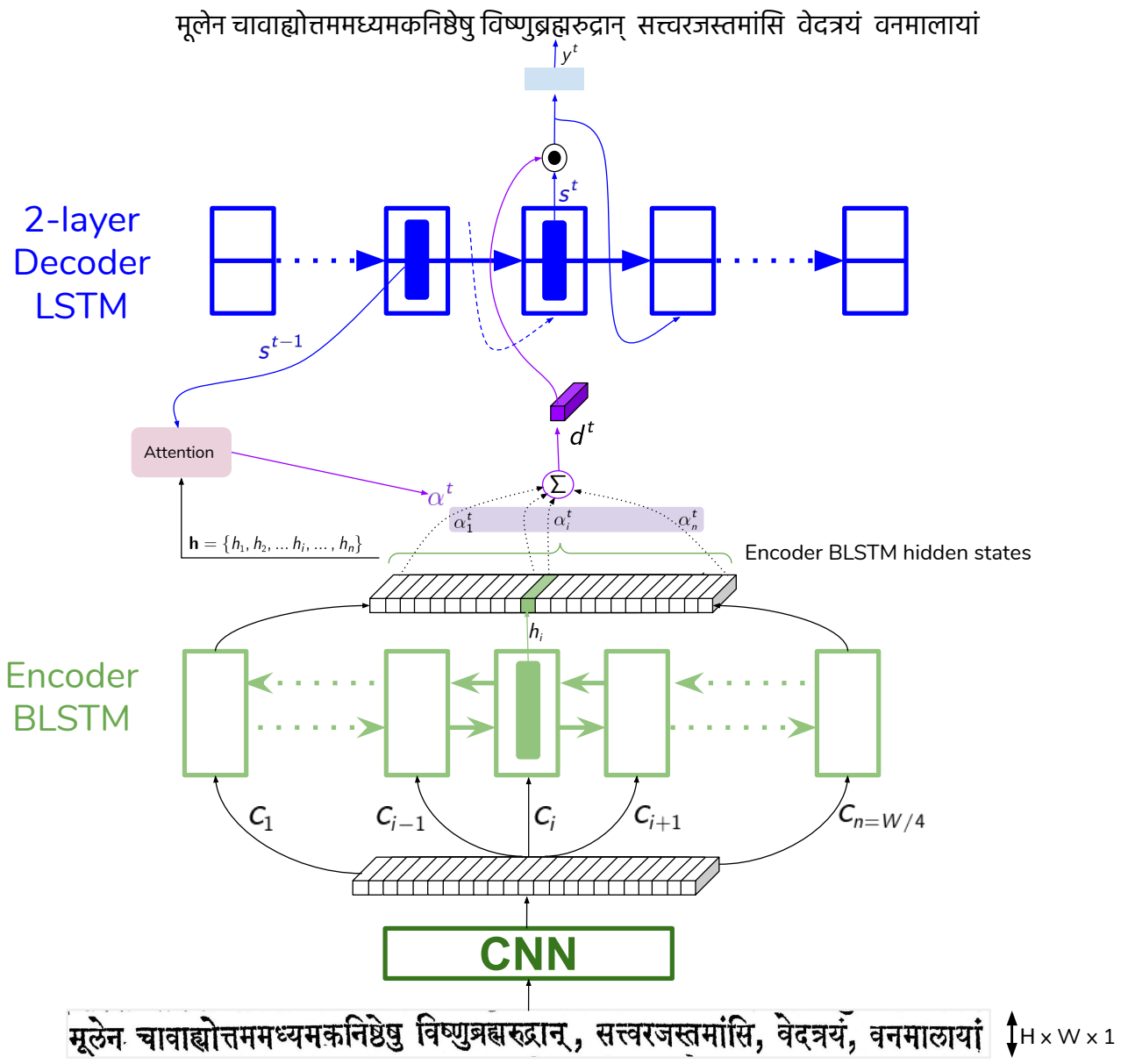

Figure 4: The architecture for our single-headed Attention LSTM OCR. In our setting, input image dimensions are $H = 32, W = 512$. The CNN architecture can be viewed in Table 2. Refer to Sec. 4 for additional details.

purple in Fig. 4) as per the equation below:

$$\alpha_i^t = softmax(v^T \sigma(W_h h_i + W_s s^{t-1})) \quad (1)$$

where v, W_h, W_s are learnable parameters and s^{t-1} is the hidden state of the Decoder LSTM from the $(t-1)$th timestep. The resulting coefficients are used to perform soft selection of encoder features $\mathbf{h}$ to obtain the so-called context vector as follows:

$$d^t = \sum_i \alpha_i^t h_i^t \quad (2)$$

The context d^t of current timestep t is concatenated with Decoder LSTM's corresponding hidden state s^t and mapped to a linear layer followed by a softmax activation to obtain the final output y^t. Note that the result of concatenation mentioned above is fed as input to next timestep of the Decoder.

Layer	Filter Size (W, H)	Filters	Stride (W, H)	Output Dims. (W, H, D)
Conv1	(3, 3)	64	(1, 1)	(32, 512, 64)
ReLU	-	-	-	(32, 512, 64)
MaxPool1	(2, 2)	-	(2, 2)	(16, 256, 64)
Conv2	(3, 3)	128	(1, 1)	(16, 256, 128)
ReLU	-	-	-	(16, 256, 128)
MaxPool2	(2, 2)	-	(2, 2)	(8, 128, 128)
Conv3	(3, 3)	256	(1, 1)	(8, 128, 256)
BatchNorm	-	-	-	(8, 128, 256)
ReLU	-	-	-	(8, 128, 256)
Conv4	(3, 3)	256	(1, 1)	(8, 128, 256)
ReLU	-	-	-	(8, 128, 256)
MaxPool3	(2, 1)	-	(2, 1)	(4, 128, 256)
Conv5	(3, 3)	512	(1, 1)	(4, 128, 512)
BatchNorm	-	-	-	(4, 128, 512)
ReLU	-	-	-	(4, 128, 512)
Conv6	(3, 3)	512	(1, 1)	(4, 128, 512)
ReLU	-	-	-	(4, 128, 512)
MaxPool4	(2, 1)	-	(2, 1)	(2, 128, 512)
Conv7	(2, 2)	512	(1, 1)	(2, 128, 512)
BatchNorm	-	-	-	(2, 128, 512)
ReLU	-	-	-	(2, 128, 512)
MaxPool5	(2, 1)	-	(2, 1)	(1, 128, 512)
Dropout(0.5)	-	-	-	(1, 128, 512)

Table 2: CNN details. W, H and D represent width, height and depth respectively.

4.1. Optimization

The architecture is trained end-to-end using cross-entropy loss defined with respect to the line sequence. For optimization, we use AdaDelta with decay rate of 0.95 which changes the learning rate based on a moving window of gradient updates instead of accumulating all past gradients. The initial learning rate is set to 1.0. We use mini-batches of size 16 and stop training after $310k$ iterations. The training takes 40 hours on a RTX 2080i GPU.

5. Experiments and Results

We use 70% of our real data for training, 10% for validation and remaining 20% for testing the models. We train our model (Attention LSTM) with different training configurations - C1 (mix training): mixture of synthetic and real data, C2 (synthetic training, real fine-tune): pre-training on synthetic data and fine-tuning on real data [5], C3 (mix training, real fine-tune): pre-training on mixture of synthetic and real data and then fine-tuning on real data. We use the hybrid CNN-RNN model introduced by Shi *et. al.* [32] as a baseline. Additionally, we also compare our performance with Tesseract [36] and the commerical OCR system Ind.Senz [10]. We use the standard Character Error Rate (CER) and Word Error Rate (WER) as performance measures.

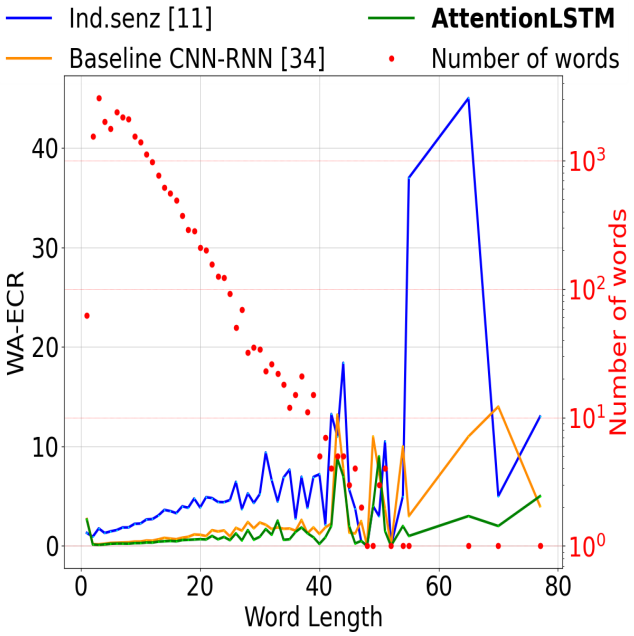

Figure 5: Distribution of word-averaged erroneous character rate (WA-ECR) as a function of length, for different models. The lower WA-ECR the better. The test words histogram in terms of word lengths can also be seen in the plot (red dots, log scale). See Section 5 for details.

Row	Model	Training Config	CER (%)	WER (%)
1	Attention LSTM	C3:mix training + real finetune	**3.71**	**15.97**
2	Attention LSTM	C2:synth. training + real finetune	4.59	17.97
3	Attention LSTM	C1:mix training	5.18	19.14
4	CNN-RNN [32]	C3:mix training + real finetune	4.72	22.53
5	CNN-RNN [32]	C2:synth. training + real finetune	4.78	22.97
6	CNN-RNN [32]	C1:mix training	4.83	23.04
7	Ind.senz [10]	–	25.66	40.81
8	Tesseract [36]	–	17.96	69.89

Table 3: Character Error Rates (CER) and Word Character Error Rates (WER) for various models - smaller is better. Our Attention LSTM model achieves better performance compared to baselines.

The results can be viewed in Table 3. As the first three rows show, our Attention LSTM model performs better than the competing baseline (CNN-RNN) – our model decreases WER by 7.07 additional points compared to CNN-RNN. Our model performs substantially better than Ind.senz, decreasing the WER by 24.84 in this case. Also, we find the C3 training configuration mentioned above to be the most beneficial for performance, not only for our model, but also for the competing CNN-RNN baseline.

To gain a better insight into performance, we compute the word-averaged erroneous character rate (WA-ECR). This is defined as follows: Consider a word length l. Suppose e_l is the number of erroneous characters across all words of length l and n_l is the number of l-length words in

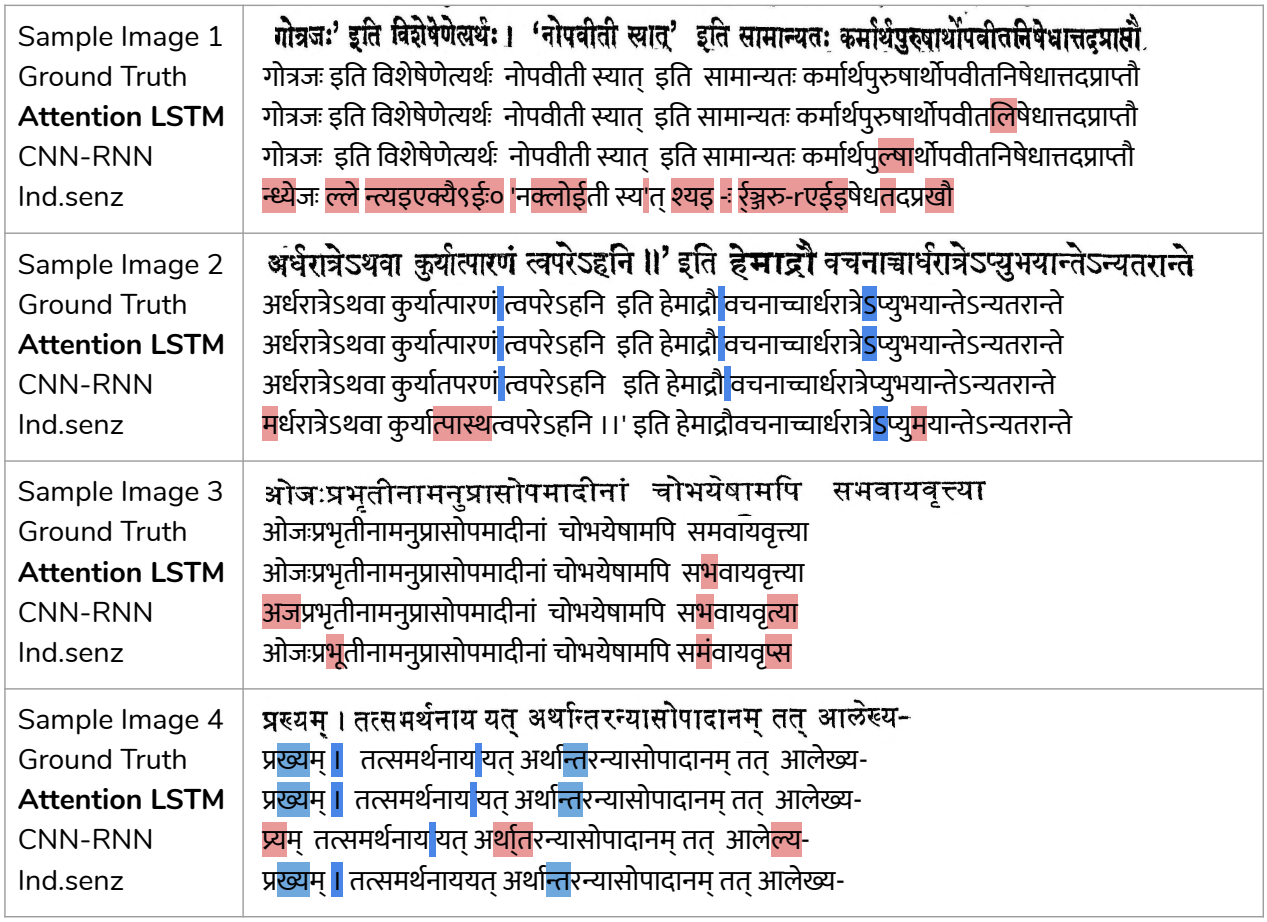

Figure 6: Qualitative results for different models. Errors relative to ground truth are highlighted in red. Blue highlighting indicates text missing from at least one of the OCRs. A larger amount of blue within a line for an OCR indicates better coverage relative to others OCRs. Smaller amount of red indicates absence of errors.

test set. WA-ECR$_l = e_l/n_l$. Given the heavy-tailed nature of our data distribution, this is a strict performance measure which heavily penalizes models which exhibit large number of word-level errors for longer words. As can be seen in Figure 5, our model has low WA-ECR across the range of word lengths. Although the baseline CNN-RNN also has a low WA-ECR, our model's WA-ECR plot is lower than the baseline's, especially for word lengths above 40. This demonstrates the importance of our model's attention mechanism in recognizing especially long out-of-vocabulary words. Also, note that our model's WA-ECR slightly increases for very long words. This is due to the relative scarcity of large length words in the real datasets used for training the OCR models (see Figure 3). This is an issue we wish to address in future work.

In our preliminary experiments, we considered pre-training our model solely on synthetic data. However, this by itself, did not seem useful – the resulting model had a CER of 27.1 and a WER of 76.01. However, as the other rows in Table 3 for our model show, the utility of synthetic data lies in its ability to provide a good starting point for training with real data.

The qualitative results comparing our model with baselines can be viewed in Figure 6. We highlight errors in red. If a particular OCR (say OCR-x) is successful in recognizing a piece of text but some of the competing OCRs fail to contain this text, we blue highlight the corresponding region in text output of OCR-x. The corresponding blue highlight region can also be present in ground-truth. Note that higher the blue region in OCR text, better the OCR (w.r.t other OCR systems which do not highlight it). In summary, absence of red and large presence of blue indicate the relative superiority of the OCR. The qualitative results once again emphasize the good performance of our Attention LSTM approach and reinforce observations made earlier.

6. Conclusion

We presented a single-headed Attention LSTM OCR for the challenging domain of classical Indic documents containing arbitrarily long words. Our model achieves a low WER on a challenging dataset of Indic documents, outperforming challenging baselines. Our design choice of using attention mechanism, coupled with a careful training regime, play a significant role in achieving good performance. As part of this effort, we also introduced a dataset of 23848 annotated line images. Additionally, we also created a synthetic image dataset which significantly aids the overall OCR development process. Our work sets the stage for development of high-performing OCRs which can be fruitfully applied to the numerous classical Indic document collections that exist. As future work, we wish to refine the Attention LSTM model with a focus to further decrease WER, incorporate OCR correction systems [27] and also to decrease the reliance on real data for training.

7. Acknowledgements

We would like to thank Dr. Sai Susarla from MIT-SVS (https://mitvedicsciences.edu.in/), for providing the annotated data. We would also like to thank the maintainer(s) of https://sanskritdocuments.org.

References

[1] Meduri Avadesh and Navneet Goyal. Optical Character Recognition for Sanskrit Using Convolution Neural Networks. In *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*, pages 447–452. IEEE, 2018. 1, 2

[2] Veena Bansal and RMK Sinha. Integrating Knowledge Sources in Devanagari Text Recognition System. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 30(4):500–505, 2000. 2

[3] Bhattoji and Sriśa Chandra Vasu. *The Siddhanta Kaumudi of Bhattoji Diksita*. Motilal Banarsidass, 1906. 1

[4] Chandan Biswas, Partha Sarathi Mukherjee, Koyel Ghosh, Ujjwal Bhattacharya, and Swapan K Parui. A Hybrid Deep Architecture for Robust Recognition of Text Lines of Degraded Printed Documents. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 3174–3179. IEEE, 2018. 1, 2

[5] Kartik Dutta, Praveen Krishnan, Minesh Mathew, and CV Jawahar. Offline Handwriting Recognition on Devanagari using a new Benchmark Dataset. In *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*, pages 25–30. IEEE, 2018. 3, 5

[6] FontHindi. Top 15 handwriting style Devanagari script fonts. http://fonthindi.blogspot.com/2014/07/top-handwriting-style-devanagari-fonts.html. Last accessed on March 13, 2020. 3

[7] Dipankar Ganguly, Sumeet Agarwal, and Santanu Chaudhury. Improving classical ocrs for brahmic scripts using script grammar learning. *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, 07:37–41, 2017. 1

[8] Google. Devanagari Fonts. https://fonts.google.com/?subset=devanagari. Last accessed on March 13, 2020. 3

[9] Script Grammar. For Indian Languages. http://language.worldofcomputing.net/grammar/script-grammar.html. Accessed March 26 2020. 1

[10] Oliver Hellwig. Indsenz OCR. *Retrieved from http://www.indsenz.com/*, 2020. Last accessed on 26 March. 2, 5

[11] CV Jawahar, MNSSK Pavan Kumar, and SS Ravi Kiran. A Bilingual OCR for Hindi-Telugu Documents and its Applications. In *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.*, pages 408–412. IEEE, 2003. 2

[12] Krishnamachari Jayanthi, Akihiro Suzuki, Hiroshi Kanai, Yoshiyuki Kawazoe, Masayuki Kimura, and Keniti Kido. Devanagari Character Recognition Using Structure Analysis. In *Fourth IEEE Region 10 International Conference TENCON*, pages 363–366. IEEE, 1989. 2

[13] Tushar Karayil, Adnan Ul-Hasan, and Thomas M Breuel. A Segmentation-Free Approach for Printed Devanagari Script Recognition. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 946–950. IEEE, 2015. 2

[14] Sivan Keret, Lior Wolf, Nachum Dershowitz, Eric Werner, Orna Almogi, and Dorji Wangchuk. Transductive Learning for Reading Handwritten Tibetan Manuscripts. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 214–221. IEEE, 2019. 1, 2

[15] Suryaprakash Kompalli, Srirangaraj Setlur, and Venu Govindaraju. Design and Comparison of Segmentation Driven and Recognition Driven Devanagari OCR. In *Second International Conference on Document Image Analysis for Libraries (DIAL'06)*, pages 7–pp. IEEE, 2006. 2

[16] Praveen Krishnan, Naveen Sankaran, Ajeet Kumar Singh, and CV Jawahar. Towards a Robust OCR System for Indic Scripts. In *2014 11th IAPR International Workshop on Document Analysis Systems*, pages 141–145. IEEE, 2014. 1, 2

[17] Chen-Yu Lee and Simon Osindero. Recursive Recurrent Nets with Attention Modeling for OCR in the Wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2231–2239, 2016. 3

[18] Nam Tuan Ly, Cuong Tuan Nguyen, and Masaki Nakagawa. An Attention-Based End-to-End Model for Multiple Text Lines Recognition in Japanese Historical Documents. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 629–634. IEEE, 2019. 1

[19] Minesh Mathew, Ajeet Kumar Singh, and CV Jawahar. Multilingual OCR for Indic scripts. In *2016 12th IAPR Workshop on Document Analysis Systems (DAS)*, pages 186–191. IEEE, 2016. 1, 2

[20] Honey Mehta, Sanjay Singla, and Aarti Mahajan. Optical Character Recognition (OCR) system for Roman Script & English Language using Artificial Neural Network (ANN) Classifier. In *2016 International Conference on Research Advances in Integrated Navigation Systems (RAINS)*, pages 1–5. IEEE, 2016. 1

[21] Thomas Milo and Alicia González Martínez. A New Strategy for Arabic OCR: Archigraphemes, Letter Blocks, Script grammar, and shape synthesis. In *Proceedings of the 3rd International Conference on Digital Access to Textual Cultural Heritage*, pages 93–96, 2019. 1

[22] Marcin Namysl and Iuliu Konya. Efficient, Lexicon-Free OCR using Deep Learning. *arXiv preprint arXiv:1906.01969*, 2019. 1

[23] Premkumar S Natarajan, Ehry MacRostie, and Michael Decerbo. The BBN Byblos Hindi OCR system. In *Document Recognition and Retrieval XII*, volume 5676, pages 10–16. International Society for Optics and Photonics, 2005. 1

[24] Debabrata Paul and Bidyut Baran Chaudhuri. A BLSTM Network for Printed Bengali OCR System with High Accuracy. *arXiv preprint arXiv:1908.08674*, 2019. 1, 2

[25] John C Platt, Nello Cristianini, and John Shawe-Taylor. Large margin dags for multiclass classification. In *Advances in neural information processing systems*, pages 547–553, 2000. 2

[26] Shalini Puri and Satya Prakash Singh. An efficient Devanagari character classification in printed and handwritten documents using SVM. *Procedia Computer Science*, 152:111–121, 2019. 2

[27] Rohit Saluja, Devaraj Adiga, Parag Chaudhuri, Ganesh Ramakrishnan, and Mark Carman. Error detection and corrections in indic ocr using lstms. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 17–22. IEEE, 2017. 3, 7

[28] Naveen Sankaran and CV Jawahar. Recognition of printed Devanagari text using BLSTM Neural Network. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pages 322–325. IEEE, 2012. 1, 2

[29] Naveen Sankaran, Aman Neelappa, and CV Jawahar. Devanagari Text Recognition: A Transcription Based Formulation. In *2013 12th International Conference on Document Analysis and Recognition*, pages 678–682. IEEE, 2013. 1, 2

[30] Ishwar K Sethi and B Chatterjee. Machine recognition of constrained hand printed devanagari. *Pattern recognition*, 9(2):69–75, 1977. 2

[31] Richa Sharma and Tarun Mudgal. Primitive Feature-Based Optical Character Recognition of the Devanagari Script. In *Progress in Advanced Computing and Intelligent Engineering*, pages 249–259. Springer, 2019. 2

[32] Baoguang Shi, Xiang Bai, and Cong Yao. An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition. *IEEE transactions on pattern analysis and machine intelligence*, 39(11):2298–2304, 2016. 5

[33] Shobhika. A Devanāgarī font for scholars. https://github.com/Sandhi-IITBombay/Shobhika. Last accessed on March 13, 2020. 3

[34] Raghuraj Singh, CS Yadav, Prabhat Verma, and Vibhash Yadav. Optical character recognition (OCR) for printed devnagari script using artificial neural network. *International Journal of Computer Science & Communication*, 1(1):91–95, 2010. 1

[35] RMK Sinha and HN Mahabala. Machine Recognition of Devanagari Script. *IEEE Transactions on Systems, Man and Cybernetics*, 9(8):435–441, 1979. 2

[36] Ray Smith. Tesseract-OCR. *Retrieved from https://github.com/tesseract-ocr/*, 2020. Accessed 26 March 2020. 2, 5

[37] India Typing. Download 50 Beautiful Unicode Hindi Fonts. http://indiatyping.com/index.php/download/top-50-hindi-unicode-fonts-free. Last accessed on March 13, 2020. 3

[38] Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. Grammar as a foreign language. In *Advances in neural information processing systems*, pages 2773–2781, 2015. 3

[39] Ziang Yan, Chengzhe Yan, and Changshui Zhang. Rare Chinese character recognition by Radical extraction network. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 924–929. IEEE, 2017. 1