Upcycle Your OCR: Reusing OCRs for Post-OCR Text Correction in Romanised Sanskrit

Amrith Krishna[#], Bodhisattwa Prasad Majumder[*], Rajesh Shreedhar Bhat[**], and Pawan Goyal[#]

[#]Dept. of Computer Science and Engineering, IIT Kharagpur,
[*]Dept. of Computer Science, University of California, San Diego
[**]Walmart Labs, India
amrith@iitkgp.ac.in, bmajumde@eng.ucsd.edu,
rajeshbhatpesit@gmail.com, pawang@cse.iitkgp.ernet.in

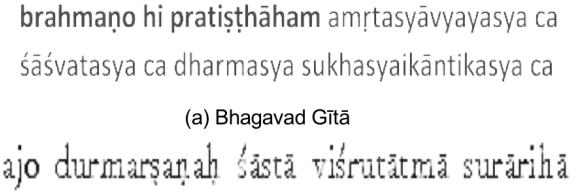
Abstract

We propose a post-OCR text correction approach for digitising texts in Romanised Sanskrit. Owing to the lack of resources our approach uses OCR models trained for other languages written in Roman. Currently, there exists no dataset available for Romanised Sanskrit OCR. So, we bootstrap a dataset of 430 images, scanned in two different settings and their corresponding ground truth. For training, we synthetically generate training images for both the settings. We find that the use of copying mechanism (Gu et al., 2016) yields a percentage increase of 7.69 in Character Recognition Rate (CRR) than the current state of the art model in solving monotone sequence-to-sequence tasks (Schnober et al., 2016). We find that our system is robust in combating OCR-prone errors, as it obtains a CRR of 87.01% from an OCR output with CRR of 35.76% for one of the dataset settings. A human judgement survey performed on the models shows that our proposed model results in predictions which are faster to comprehend and faster to improve for a human than the other systems[1].

1 Introduction

Sanskrit used to be the 'lingua franca' for the scientific and philosophical discourse in ancient India with literature that spans more than 3 millennia. Sanskrit primarily had an oral tradition, and the script used for writing Sanskrit varied widely across the time spans and regions. With the advent of printing press, Devanagari emerged as the prominent script for representing Sanskrit. With standardisation of Romanisation using IAST in 1894 (Monier-Williams, 1899), printing in Sanskrit was extended to roman scripts as well. There has been a surge in digitising printed Sanskrit manuscripts written in Roman such as the ones currently digitised by the 'Krishna Path' project[2].

In this work, we propose a model for post-OCR text correction for Sanskrit written in Roman. Post-OCR text correction, which can be seen as a special case of spelling correction (Schnober et al., 2016), is the task of correcting errors that tend to appear in the output of the OCR in the process of converting an image to text. The errors incurred from OCR can be quite high due to numerous factors including typefaces, paper quality, scan quality, etc. The text can often be eroded, can contain noises and the paper can be bleached or tainted as well (Schnober et al., 2016). Figure 1 shows the sample images we have collected for the task. Hence it is beneficial to perform a post-processing on the OCR output to obtain an improved text.



(a) Bhagavad Gītā

(b) Sahaśranāma

Figure 1: Sample images from our test set with different stylistic parameters

In the case of Indic OCRs, there have been considerable efforts in collection and annotation of data pertaining to Indic Scripts (Kumar and Jawahar, 2007; Bhaskarabhatla et al., 2004; Govindaraju and Setlur, 2009; Krishnan et al., 2014). Earlier attempts on Indian scripts were primarily based on handcrafted templates (Govindan and Shivaprasad, 1990; Chaudhuri and Pal, 1997) or features (Arora et al., 2010; Pal et al., 2009) which extensively used the script and language-specific

[1]The data and the codes for our system are available here - https://github.com/majumderb/sanskrit-ocr

[2]http://www.krishnapath.org/library/

information (Krishnan et al., 2014). Sequential labelling approaches were later proposed that take the word level inputs and make character level predictions (Shaw et al., 2008; Hellwig, 2015). The word based sequence labelling approaches were further extended to use neural architectures, especially using RNNs and its variants such as LSTMs and GRUs (Sankaran and Jawahar, 2012; Krishnan et al., 2014; Saluja et al.; Adiga et al., 2018; Mathew et al., 2016). But, OCR is putative in exhibiting few long-range dependencies (Schnober et al., 2016). Singh and Jawahar (2015) find that extending the neural models to process the text at the sentence level (or a textline) leads to improvement in the performance of the OCR systems. This was further corroborated by Saluja et al. where the authors found that using words within a context window of 5 for a given input word worked particularly well for the Post-OCR text correction in Sanskrit. In the case of providing a text line as input, we are essentially providing more context about the input in comparison to the word level models and the RNN (or LSTM) cells are powerful enough to capture the long-term dependencies. Particularly for Indian languages, this decision is beyond a question of performance. In Sanskrit, the word boundaries are often obscured due to phonetic transformations at the word boundaries known as Sandhi. Word segmentation of Sanskrit constructions is a matter of research on its own (Krishna et al., 2016a; Reddy et al., 2018). However, none of the existing systems are equipped for incorrect spellings and hence these systems may be brittle (Belinkov and Bisk, 2018) when it comes to handling spelling variations in the input. Hence, in our case, we assume an unsegmented sequence as our input and then we perform our Post-OCR text correction on the text. We hypothesise that this will improve the segmentation process and other downstream tasks for Sanskrit in a typical NLP pipeline.

Our major contributions are:

1. Contrary to what is observed in Schnober et al. (2016), an encoder-decoder model, when equipped with copying mechanism (Gu et al., 2016), can outperform a traditional sequence labelling model in a monotone sequence labelling task. Our model outperforms Schnober et al. (2016) in the Post-OCR text correction for Romanised Sanskrit task by 7.69 % in terms of CRR.

2. By making use of digitised Sanskrit texts, we generate images as synthetic training data for our models. We systematically incorporate various distortions to those images so as to emulate the settings of the original images.

3. Through a human judgement experiment, we asked the participants to correct the mistakes from a predicted output from the competing systems. We find that participants were able to correct predictions from our system more frequently and the corrections were done much faster than the CRF model by Schnober et al. (2016). We observe that predictions from our model score high on acceptability (Lau et al., 2015) than other methods as well.

2 Model Architecture

In principle, the output from any OCR which recognises Romanised Sanskrit can be used as the input to our model. Currently, there exist limited options for recognising Romanised Sanskrit texts from scanned documents. Possibly, the commercial OCR offering by Google as part of their proprietary cloud vision API and SanskritOCR[3] might be the only two viable options. SanskritOCR provides an online interface to the Tesseract OCR, an open source multilingual OCR (Smith, 2007; Smith et al., 2009; Smith, 1987), trained specifically for recognising Romanised Sanskrit. Additionally, we trained an offline version of Tesseract to recognise the graphemes in the Romanised Sanskrit alphabet. In both the models we find that many scanned images, especially similar to the one shown in Figure 1b, were not recognised by the system. We hypothesise this to be due to lack of enough font styles available in our collection, in spite of using a site with the richest collection of Sanskrit fonts[4]. This leaves the Google OCR as the only option.

Considering the fact that working with a commercial offering from Google OCR may not be an affordable option for various digitisation projects, we chose to use Tesseract with models trained for other languages written in Roman script. All the Latin or Roman scripts in the pre-trained models

[3] https://sri.auroville.org/projects/sanskrit-ocr/. It provides interface to tesseract and Google OCR as well.

[4] More details about the training procedure in §1 of the supplementary material

of Tesseract are trained on 400,000 text-lines spanning about 4500 fonts[5].

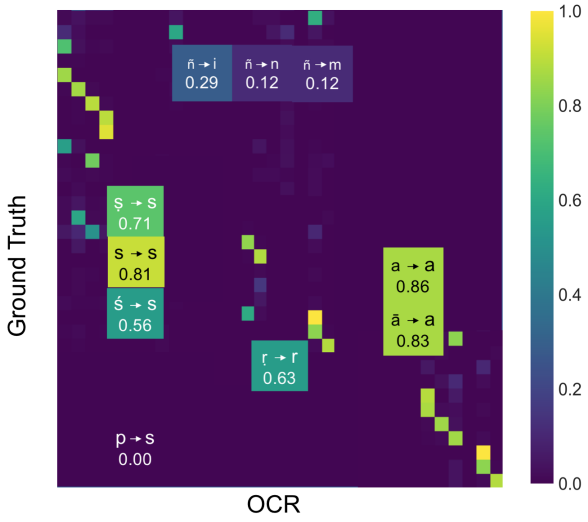


Figure 2: Heatmap of occurrences of majorly confusing character pairs between Ground Truth and OCR

Use of OCR with pre-trained models for other languages French alphabet has the highest grapheme overlap with that of the Sanskrit alphabet (37 of 50), while all other languages have one less grapheme common with Sanskrit. Hence, we arbitrarily take 5 of the languages in addition to French and perform our analysis. Table 1 shows the character recognition rate (CRR) for OCR using alphabets of different languages, when performed on a dataset of 430 scanned images (§3.1). The table also shows the count of error types made by the OCR after alignment (Jiampojamarn et al., 2007; D'hondt et al., 2016). All the languages have a near similar CRR with English and French leading the list. Based on our observations on the OCR performance, we select English for our further experiments.

Upcycling such a pre-trained model brings its own challenges. For instance, the missing 14 Sanskrit graphemes[6] in English are naturally mispredicted to other graphemes. This leads to ambiguity as the correct and the mispredicted characters now share the same target. Figure 2 shows the heat-map for such mis-predictions when we used the OCR on the set of 430 scanned images. Here, we zoom the relevant cases and show the row-normalised proportion of predictions[7].

[5] https://github.com/tesseract-ocr/tesseract/wiki/TrainingTesseract-4.00

[6] Detailed in §2 of the Supplementary Material

2.1 System Descriptions

We formalise the task as a monotone seq2seq model. We use an encoder-decoder framework that takes in a character sequence as input and the model finds embeddings at a sub-word level both at the encoder and decoder side. Here the OCR output forms input to the model. Keeping the task in mind we make two design decisions for the model. One is the use of copying mechanism (Gu et al., 2016) and other is the use of Byte Pair Encoding (BPE) (Sennrich et al., 2016) to learn a new vocabulary for the model.

CopyNet (Gu et al., 2016): Since it is possible that there will be reasonable overlap between the input and output strings, we use the copying mechanism as mentioned in CopyNet (Gu et al., 2016). The model essentially learns two probability distributions, one for generating an entry at the decoder and the other for copying the entry from the encoder. The final prediction is based on the sum of both the probabilities for the class. Given an input sequence $\boldsymbol{X} = (\boldsymbol{x_1}, ..., \boldsymbol{x_N})$ we define $\mathcal{X}$, for all the *unique* entries in the input sequence. We also define the vocabulary $\mathcal{V} = \{v_1, ..., v_N\}$. Let the out-of-vocabulary (OOV) words be represented with UNK. The probability of the generate mode g and copy mode c are given by

$$p(\boldsymbol{y_t}, \mathsf{g}|\cdot) = \begin{cases} \frac{1}{Z} e^{\psi_g(\boldsymbol{y_t})}, & \boldsymbol{y_t} \in \mathcal{V} \\ 0, & \boldsymbol{y_t} \in \mathcal{X} - \mathcal{V} \\ \frac{1}{Z} e^{\psi_g(\text{UNK})} & \boldsymbol{y_t} \notin \mathcal{V} \cup \mathcal{X} \end{cases}$$

$$p(\boldsymbol{y_t}, \mathsf{c}|\cdot) = \begin{cases} \frac{1}{Z} \sum_{j:\boldsymbol{x_j}=\boldsymbol{y_t}} e^{\psi_c(\boldsymbol{x_j})}, & \boldsymbol{y_t} \in \mathcal{X} \\ 0 & \text{otherwise} \end{cases}$$

where $\psi_g(\cdot)$ and $\psi_c(\cdot)$ are score functions for generate-mode (g) and copy-mode (c), respectively, and Z is the normalization term shared by the two modes, $Z = \sum_{v \in \mathcal{V} \cup \{\text{UNK}\}} e^{\psi_g(v)} + \sum_{\boldsymbol{x} \in \boldsymbol{X}} e^{\psi_c(\boldsymbol{x})}$. The scoring function for both the modes, respectively, are

$$\psi_g(\boldsymbol{y_t} = v_i) = \boldsymbol{v}_i^\top \boldsymbol{W_o} \boldsymbol{s_t}, \quad v_i \in \mathcal{V} \cup \text{UNK}$$

$$\psi_c(\boldsymbol{y_t} = \boldsymbol{x_j}) = \sigma\left(\mathbf{h}_j^\top \mathbf{W}_c\right) \mathbf{s}_t, \quad \boldsymbol{x_j} \in \mathcal{X}$$

where $\mathbf{W}_c \in \mathbb{R}^{d_h \times d_s}$, and σ is a non-linear activation function (Gu et al., 2016).

[7] A more detailed figure with all the cases are available in the supplementary material in §3.

Language	Bhagavad Gĭtā				Sahaśranāma				Combined
	CRR	Ins	Del	Sub	CRR	Ins	Del	Sub	CRR
English	84.92	23	63	1868	64.06	73	696	1596	80.08
French	84.90	21	102	1710	63.91	91	702	1670	80.04
Finnish	82.61	15	141	1902	61.31	80	730	1821	78.81
Italian	83.45	20	73	1821	62.19	84	690	1673	79.03
Irish	84.52	12	78	1810	63.81	72	709	1841	79.93
German	84.40	33	72	1821	63.79	87	723	1874	79.12

Table 1: OCR performances for different languages with overall CRR, total Insertion, Deletion and Substitution errors.

BPE (Sennrich et al., 2016) : Sanskrit is a morphologically rich language. A noun in Sanskrit can have 72 different inflections and a verb may have more than 90 inflections. Additionally, Sanskrit corpora generally express a compound rich vocabulary (Krishna et al., 2016b). Hence, in a typical Sanskrit corpus, the majority of the tokens appear less than 5 times (§3.1). These are generally considered to be rare words in a corpus (Sennrich et al., 2016). However, corpora dominated by rare words are difficult to handle for a statistical model like ours. To combat the sparsity of the data, we convert the tokens into sub-word n-grams using Byte Pair Encoding (BPE) (Sennrich et al., 2016). Methods such as wordpiece (Schuster and Nakajima, 2012) as well as Sennrich et al. (2016) are means of obtaining a new vocabulary for a given corpus. Every sequence in the corpus is then re-written as a sequence of tokens in terms of the sub-word units which forms the type in the new vocabulary so obtained. These methods essentially use a data-driven approach to maximise the language-model likelihood of the training data, given an evolving word definition (Wu et al., 2016).

We explicitly set the minimum count for a token in the new vocabulary to appear in the corpora as 30. We learn a new vocabulary of size 82 with 22 of them having a length 1 and the rest with a length 2. The IAST standardisation of the Romanised Sanskrit contains 50 graphemes in Sanskrit alphabet. About 12 of the graphemes are represented using 2 character roman character combinations. Now, in the vocabulary learnt using BPE, 7 of the graphemes were not present. Hence, we add them in addition to the 82 entries learnt as vocabulary. This makes the total vocabulary to be 89. By using the new vocabulary, it is guaranteed that there will be no Out Of Vocabulary (OOV) words in our model.

We use 3 stacked layers of LSTM at the encoder and the decoder with the same settings as in Bahdanau et al. (2015). To enable copying, we share the embeddings of the source and the target vocabulary. By eliminating OOV, we make sure that copying always happens by virtue of the evidence from the training data and not by the presence of an OOV word.

3 Experiments

3.1 Dataset

Sanskrit is a low-resource language. It is extremely scarce to obtain datasets with scanned images and the corresponding aligned texts for Romanised Sanskrit. We obtain 430 scanned images as shown in Figure 1 and manually annotate the corresponding text. We use this as our test dataset, henceforth to be referred to as *OCRTest*. For training, we synthetically generate images from digitised Sanskrit texts and use them as our training set and development set. The images for training, *OCRTrain*, were generated by synthetically adding distortions to those images to match the settings of the real scanned documents.

OCRTest contains 430 images from 1) scanned copy of Vishnu Sahaśranāma[8] and 2) scanned copy of Bhagavad Gītā, a sample of each is shown in Figure 1a and 1b. 140 out of these 430 are from Sahaśranāma and the remaining are from Bhagavad Gītā.

OCRTrain: Similar to Ul-Hasan and Breuel (2013), we synthetically generate the images, which are then fed to the OCR, to obtain our training data. We use the digitised text from Śrīmad Bhāgavatam[9] for generating the synthetic images. The text contains about 14,094 verses in total, divided into 50,971 text-lines. The dataset is divided into 80-20 split as training set and development set, respectively. The corpus contains a vocabulary of 52,882 word types. 48,249 of the word types

[8] http://kirtimukha.com/
[9] https://www.vedabase.com/en/sb

in the vocabulary appear less than or equal to 5 times, of which 32,411 appear exactly once. This is primarily due to the inflectional nature of Sanskrit. We find similar trends in the vocabulary of Rāmāyaṇa[10] and Digital Corpus of Sanskrit (Hellwig, 2010-2016) as well.

3.2 Synthetic Generation of training set

Using the text-lines from Bhāgavatam, we generate synthetic images using ImageMagick[11]. The images were generated with a quality of 60 Dots Per Inch (DPI). The number of pixels along the height for each textline was kept constant at 65 pixels. We add several distortions to the synthetically generated images so as to visually match with the same settings as that of *OCRTest*. Previously, Ul-Hasan and Breuel (2013) used the approach of synthetically generating training data for multilingual OCR solution of theirs.

Table 2 shows the different parameters, namely, gamma correction, noise addition, use of structural kernel for erosion and perspective distortion, that we apply sequentially on the images so as to distort and degrade the images (Chen et al., 2014). We use grid search for the parameter estimation for these processes, where those parameters and the range of values experimented with are provided in Table 2. Finally, we filter 7 (out of 38,400 combinations) different configurations based on the distribution of Character Recognition Rate (CRR) across the images compared with that of the *OCRTest* using KL-divergence. Among these seven configurations, four are closer to the settings for Bhagavad Gītā and the remaining three for Sahaśranāma. Figure 3 shows the two different settings (closer to each of the source textbook) for the string *"ajo durmarṣaṇaḥ śāstā viśrutātmā surārihā"*, along with their corresponding parameter settings and KL-Divergence. Our training set contains images from all the 7 settings for each of the textline in OCRTrain[12].

Evaluation Metrics We use three different metrics for evaluating all our models. We use Character Recognition Rate (CRR) and Word Recognition Rate (WRR) averaged over each of the sentences in the 430 lines in the test dataset (Sankaran and Jawahar, 2012). CRR is the fraction of characters recognised correctly against the total number of characters in a line, whereas WRR is the fraction of words correctly recognised against the total number of words in a line. Additionally, we use a sentence level metric, called the acceptability score. The measure indicates the extent to which a sentence is permissible or acceptable to the speakers of the language (Lau et al., 2015). From Lau et al. (2015), we use the *NormLP* formulation for the task, as it is found to have a high correlation with the human judgements in evaluating acceptability. NormLP is calculated by obtaining the likelihood of a predicted sentence as per the model, and then normalising it by the likelihood of the string as per a unigram language model trained on a corpus with gold standard sentences. A negative sign is then given to the score. The higher the score, the more acceptable the sentence is.

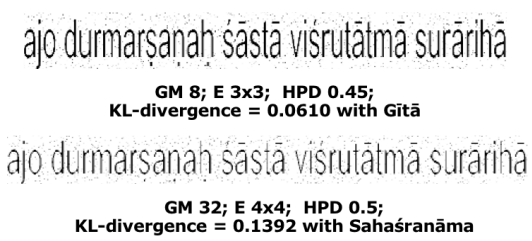


Figure 3: Samples of synthetically generated images. The parameter settings for the distortions are mentioned below the corresponding image.

3.3 Baselines

Character Tagger - Sequence Labelling using BiLSTMs This is a sequence labelling model which uses BiLSTM cells and input is a character sequence (Saluja et al.). We use categorical cross-entropy as the loss function and softmax as the activation function. For dropout, we employ spatial dropout in our architecture. The model consists of 3 layers with each layer having 128 cells. Embeddings of size 100 are randomly initialised and the learnt representations are stored in a character look-up table similar to Lample et al. (2016). In addition to every phoneme in Sanskrit as a class, we add an additional class 'no change' which signifies that the character remains as is. We also experimented with a variant where the final layer is a CRF layer (Lafferty et al., 2001). We henceforth refer to both the systems as *BiLSTM* and *BiLSTM-CRF*, respectively.

Pruned CRFs (Schnober et al., 2016): They

[10] https://sanskritdocuments.org/sites/valmikiramayan/

[11] https://www.imagemagick.org/script/index.php

[12] Samples of all the 7 seven configurations are shown in the supplementary material in §4

Process	Parameters	Range		Step size
Gamma Correction (GM)	gamma (γ)	4	64	4
Salt & Pepper Noise (SPN) (with 50% salt and 50% pepper)	percentage of pixels corrupted	0.1%	1%	0.1
Gaussian Noise (GN) (mean = 0)	standard deviation	2.5	3.5	0.25
Erosion (E) (one iteration)	kernel size (m $\times$ m)	2	5	1
Horizontal perspective distortion (HPD)	image width by image height	0.3	1	0.05

Table 2: Image pre-processing steps and parameters

are higher order CRF models (Ishikawa, 2011) that approximate the CRF objective function using coarse-to-fine decoding. Schnober et al. (2016) adapt the sequence labeling model as a seq2seq model that can handle variable length input-output pairs. Schnober et al. (2016) show that none of the neural seq2seq models considered in their work were able to outperform the Pruned CRF with order-5. The features to the model are consecutive characters within a window of size w in either of the directions of the current position at which a prediction is made. The model is designed to handle 1-to-zero and 1-to-many matches, facilitated by the use of alignment prior to training. We consider all the three settings reported in Schnober et al. (2016) and report the results for the best setting. The order-5 model which uses 6-grams within a window of 6 performs the best. Henceforth, this model is referred to as *PCRF-seq2seq* (also referred to as PCRF interchangeably).

Encoder-Decoder Models: For the seq2seq model (Sutskever et al., 2014), we use 3 stacked layers of LSTM each at the encoder and the decoder. Each layer is of 128 dimensions and weighted cross-entropy is used as the loss. We also add residual connections among the layers in a stack (Wu et al., 2016). To further capture the entire input context for making each prediction at the output, we make use of attention (Bahdanau et al., 2015), specifically Luong's attention mechanism (Luong et al., 2015). We experiment with two variants where *EncDec+Char* uses character level embeddings and *EncDec+BPE* uses embeddings with BPE.

CopyNet+BPE: The model discussed in §2. We use CopyNet+BPE and CopyNet interchangeably throughout the paper.

3.4 Results

Table 3 shows the results for all the competing systems based on the predictions from *OCRTest*. CopyNet performs the best among the competing systems across all the three metrics and on both the source texts. For the Gītā dataset, the models CopyNet and PCRF-Seq2Seq report similar performances. However, Sahaśranāma is a noisier dataset, and we find that CopyNet outperforms all other models by a huge margin. The WRR for the system is double that of the next best system (EncDec) on this dataset.

System performances for various input lengths: From Figure 4a, it can be observed that the performance in terms of CRR for CopyNet and PCRF is robust across all the lengths on strings from Gītā and never goes below 90%. For Sahaśranāma, as shown in Figure 4b, CopyNet outperforms PCRF across inputs of all the lengths except for one setting. But, in the case of WRR, CopyNet is the best performing model across all the lengths as shown in Figure 4d.

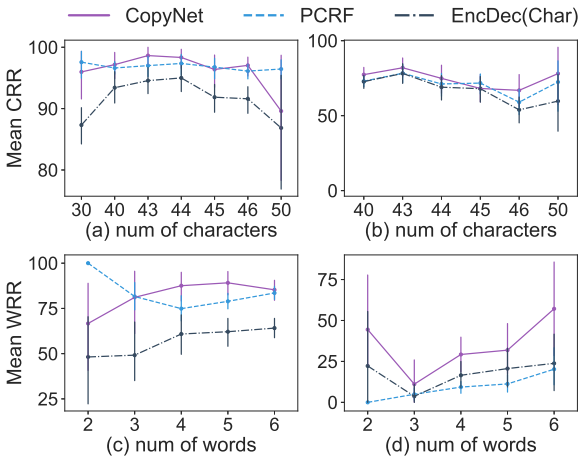


Figure 4: (a) and (b) show CRR for Gītā and Sahaśranāma respectively, for the competing systems. (c) and (d) shows WRR for Gītā and Sahaśranāma, respectively. All the entries with insufficient data-points were merged to the nearest smaller number.

Error type analysis In Table 5, we analyse the reduction in specific error types for PCRF

Model	Bhagavad Gītā			Sahaśranāma			Combined		
	CRR	WRR	Norm LP	CRR	WRR	Norm LP	CRR	WRR	Norm LP
OCR	84.81%	64.40%	–	35.76%	0.65%	–	77.88%	23.84%	–
BiLSTM	93.79%	68.60%	-0.553	61.31%	7.28%	-1.292	85.23%	45.60%	-0.852
BiLSTM-CRF	94.68%	68.60%	-0.548	65.31%	7.28%	-1.281	85.82%	45.60%	-0.847
PCRF-seq2seq	**96.87%**	70.56%	-0.227	81.77%	9.34%	-1.216	87.94%	57.17%	-0.803
EncDec+Char	91.48%	68.00%	-0.542	63.63%	15.74%	-1.321	82.51%	47.37%	-0.865
EncDec+BPE	90.92%	68.00%	-0.496	61.53%	15.74%	-1.384	83.14%	45.98%	-0.842
CopyNet+BPE	**97.01%**	**75.21%**	**-0.165**	**87.01%**	**33.47%**	**-0.856**	**89.65%**	**68.71%**	**-0.551**

Table 3: Performance in terms of CRR, WRR and Norm LP (acceptability) for all the competing models

	CRR	WRR
Bhagavad Gītā	96.80%	71.23%
Sahaśranāma	82.81%	26.01%
Combined	87.88%	60.91%

Table 4: Performance in terms of CRR, WRR for Google OCR

Model	Bhagavad Gītā			Sahaśranāma			System errors		
	Ins	Del	Sub	Ins	Del	Sub	Ins	Del	Sub
OCR	23	63	1868	73	696	1596	–	–	–
PCRF	22	57	641	72	663	932	0	73	209
CopyNet	22	**45**	629	72	**576**	561	10	5	52

Table 5: Insertion, Deletion and Substitution errors for OCR, PCRF and CopyNet modes for both the datasets. The system errors are extra errors added by the respective systems.

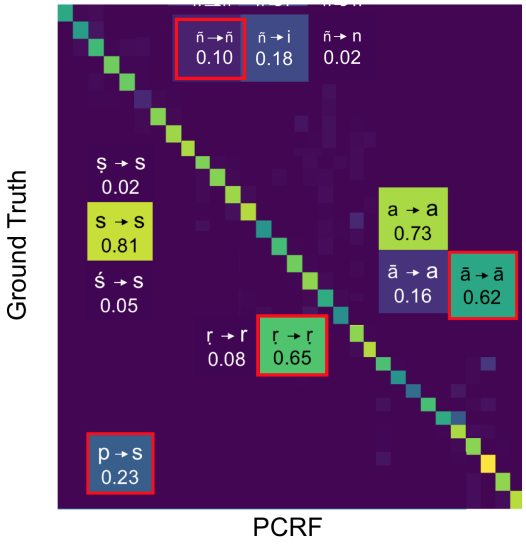
and CopyNet after the alignment of the predicted string with that of the ground truth in terms of insertion, deletion and substitution. We also report the system induced errors, where a correct component at the input (OCR output) is mispredicted to a wrong output by the model. CopyNet outperforms PCRF in correcting the errors and it also introduces lesser number of errors of its own. Both CopyNet and PCRF (Schnober et al., 2016) are seq2seq models and can handle varying length input and output. Both the systems perform well in handling substitution errors, the type which dominated the strings in *OCRTest*, though neither of the systems was able to correct the insertion errors. Insertion can be seen as a special case of 1-to-many insertion matches, which both systems are ideally capable of handling. We see that for Sahaśranāma, CopyNet corrects about 17.24 % of the deletion errors as against <5% of the deletion errors corrected by PCRF.

Since there exist 14 graphemes in Sanskrit alphabet which are not present in the English alphabet, all 14 of them get substituted to a different grapheme by the OCR. While most of them get substituted to an orthographically similar character such as ā → a and ḥ → h, we find that ñ → i does not fit the scheme, as shown in Figure 2. In the majority of the cases, CopyNet predicts them to the correct grapheme. But PCRF still fails to correct the OCR induced confusion for ñ → i in the majority of the instances. Additionally, we find that PCRF introduces its own errors, for instance it often mispredicts p → s. Figure 5 shows the overall variations in both the systems as compared to Figure 2 for OCR induced errors.

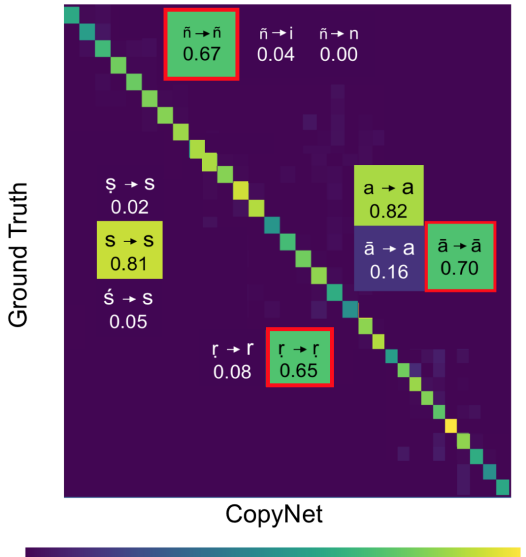
Copy or generate? For the 14 graphemes, missing at the encoder (input) but present at the decoder side during training, those predictions have to happen with high values of generate probability in general. We find that not only the average generate probability for such instances is high but also the copy probability is extremely low. For the remaining cases, we find that both generate and copy probability are higher. But it needs to be noted that the prediction is made generally by summing of both the distributions and the distributions are not complementary to each other. A similar trend can be observed in Figure 6 as well. For example in the case of a → ā, only the generate probability is high. But, for a → a, both the copy and generate probability scores are high.

Effect of BPE and alphabet in the vocabulary We further investigate the effect of our vocabulary which is the union of the alphabet in Romanised Sanskrit and what is learnt using BPE. We train the model with only the alphabet as vocabulary and find the CRR and WRR for the combined test sentences to be 86.1% and 66.09%, respectively. When using the original BPE vocabulary, we find that there is a slight increase in the performance than the current vocabulary with a CRR and WRR of 89.53% and 68.11%, respectively[13]. We also find that the current setting performs better than

[13]Please refer to §5 of the Supplementary material for the performance table



(a)



(b)

Figure 5: Heatmap for occurrences of majorly confusing character pairs between Ground Truth and predictions of (a) PCRF model (b) CopyNet model

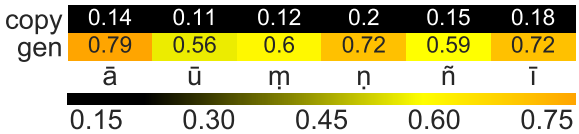


Figure 6: Heatmap of mean copy score (copy) and mean generate score (gen), respectively for 6 (of 14) graphemes not present in the English alphabet.

a model that takes word level input. The word level model shows a drop in the performance with a CRR and WRR of 86.42% and 66.54%, respectively.

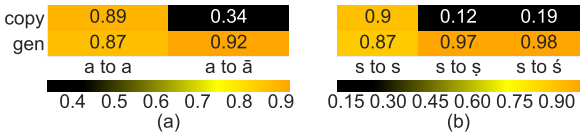


Figure 7: mean copy and generate scores for different predictions from (a) ‘a’ and (b) ‘s’.

Performance comparison to Google OCR: Google OCR is probably the only available OCR that can handle Romanised Sanskrit. We could not find the architecture of the OCR or whether the service employs post-OCR text correction. We empirically compare the performance of Google OCR on *OCRTest* with our model. Table 4 shows the results for Google OCR. Overall we find that CopyNet outperforms Google OCR across all the metrics. We find that Google OCR reports a similar CRR for Gītā with that of ours, but still reports a lower WRR than ours. The system performs better than PCRF in all the metrics other than CRR for Gītā.

Image quality: Our training set was generated with a quality of 60 DPI for the images. We generate images corresponding to strings in *OCRTrain* with DPI of 50 to 300 in step sizes of 50 for a sample of 500 images. We use noise settings as shown in Figure 3. The OCR output of the said strings remained as is with that of the one generated with a DPI of 60. This experiment can be seen as a proxy in evaluating the robustness of the OCR to various scanning qualities of input. Our choice of DPI as 60 was based on the lowest setting we observed in digitisation attempts in Sanskrit texts.

Effect of adding distortions to the synthetically generated images: Table 3 shows the system performance after training our model on data generated as per the procedure mentioned in Section 3.2. Here, we make an implicit assumption that we can have access to a sample of textline images annotated with the corresponding text from the manuscript for which the Post-OCR text correction needs to be performed. This also mandates retraining the model for every new manuscript. We attempted for a more generalised version of our model, by using training data where the image generation settings are not inspired from the target manuscript for which the task needs to be performed. Using the settings from (Chen et al., 2014) for inducing noise, we generated 10 random noise configurations. Here the step sizes were

fixed at values such that each parameter, except erosion (E), can assume 5 values each uniformly spread across the corresponding ranges considered. From a total of 2500 ($5\times5\times5\times5\times4$) configuration options, 10 random settings were chosen. Every textline was generated with each of the 10 different settings. The resulting model using CopyNet produced a CRR of 89.02% (96.99% for Gītā and 85.62% for Sahaśranāma) on the test set, which is close to the reported CRR of 89.65 in Table 3. The noise ranges chosen are used directly from (Chen et al., 2014) and are not influenced by the test data in hand.

We also experimented with a setting where no noise was added to the synthetically generated images and the images were fed to the OCR. We obtained a CRR of 80.12% from OCR, where the errors arose mostly from the missing graphemes in the alphabet getting mispredicted to a different grapheme. CopyNet after training with the text so generated reported a CRR of 86.81% (96.01% for Gītā, 75.78% for Sahaśranāma) on the test data.

Human judgement survey: In this survey[14], we evaluate how often a human can recognise the correct construction by viewing only the prediction from one of the systems. We also evaluate how fast a human can correct them. We selected 15 constructions from Sahaśranāma, and obtained the system outputs from the OCR, CopyNet and PCRF for each of these. The average length of a sentence is 41.73 characters, all ranging between 23 and 47 characters. A respondent is shown a system prediction (system identity anonymised) and is asked to type the corrected string without referring to any sources. A respondent gets 15 different strings altogether, 5 each from each of the three systems. We consider responses from 9 participants where all of them at least have an undergraduate degree in Sanskrit linguistics. Altogether from 3 sets of questionnaires, we have 45 strings (3 outputs for a given string). Every string obtained 3 impressions. We find that a participant on an average could identify 4.44 sentences out of 5 from the CopyNet, while it was only 3.56 for PCRF and 3.11 for the OCR output. The average time taken to complete the correction of a string was 81.4 seconds, 106.6 seconds and 127.6 seconds for CopyNet, PCRF and OCR, respectively.

[14]More details at §6 of Supplementary material

4 Conclusion

In this work, we proposed an OCR based solution for digitising Romanised Sanskrit. Our work acts as a Post-OCR text correction approach and is devoid of any OCR-specific feature engineering. We find that the use of copying mechanism in encoder-decoder performs significantly better than other seq2seq models for the task. Our model outperforms the commercially available Google OCR on the Sahaśranāma texts. From our experiments, we find that CopyNet performs stably even for OCR outputs with a CRR as low as 36%. Our immediate research direction will be to rectify insertion errors which currently are not properly handled. Also, there are 135 languages which directly share the Roman alphabet but only 35 of them have OCR system available. Our approach can be easily extended to provide a post-processed OCR for those languages.

Acknowledgements

We are grateful to Amba Kulkarni, Arnab Bhattacharya, Ganesh Ramakrishnan, Rohit Saluja, Devaraj Adiga and Hrishikesh Terdalkar for helpful comments and discussions related to Indic OCRs. We would like to thank Madhusoodan Pai, Sanjeev Panchal, Ganesh Iyer and his students for helping us with the human judgement survey. We thank the anonymous reviewers for their constructive and helpful comments, which greatly improved the paper.

References

Devaraj Adiga, Rohit Saluja, Vaibhav Agrawal, Ganesh Ramakrishnan, Parag Chaudhuri, K Ramasubramaniam, and Malhar Kulkarni. 2018. Improving the learnability of classifiers for sanskrit ocr corrections. In *The 17th World Sanskrit Conference, Vancouver, Canada*. IASS.

Sandhya Arora, Debotosh Bhattacharjee, Mita Nasipuri, Dipak Kumar Basu, and Mahantapas Kundu. 2010. Recognition of non-compound handwritten devnagari characters using a combination of mlp and minimum edit distance. *International Journal of Industrial Electronics and Electrical Engineering*.

Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the Third International Conference on Learning Representation (ICLR)*, San Diego, CA, USA.

Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and natural noise both break neural machine translation. In *The Sixth International Conference on Learning Representations (ICLR)*, New Orleans, USA.

Ajay S Bhaskarabhatla, Sriganesh Madhvanath, MNSSKP Kumar, A Balasubramanian, and CV Jawahar. 2004. Representation and annotation of online handwritten data. In *Ninth International Workshop on Frontiers in Handwriting Recognition*, pages 136–141, Tokyo, Japan. IEEE.

BB Chaudhuri and U Pal. 1997. An ocr system to read two indian language scripts: Bangla and devnagari (hindi). In *Proceedings of the Fourth International Conference on Document Analysis and Recognition*, volume 2, pages 1011–1015, Ulm, Germany. IEEE.

Guang Chen, Jianchao Yang, Hailin Jin, Jonathan Brandt, Eli Shechtman, Aseem Agarwala, and Tony X Han. 2014. Large-scale visual font recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3598–3605.

Eva D'hondt, Cyril Grouin, and Brigitte Grau. 2016. Low-resource ocr error detection and correction in french clinical texts. In *Proceedings of the Seventh International Workshop on Health Text Mining and Information Analysis*, pages 61–68, Auxtin, TX. Association for Computational Linguistics.

VK Govindan and AP Shivaprasad. 1990. Character recognitiona review. *Pattern recognition*, 23(7):671–683.

Venu Govindaraju and Srirangaraj Setlur. 2009. *Guide to OCR for Indic Scripts*. Springer.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany. Association for Computational Linguistics.

Oliver Hellwig. 2010-2016. *DCS - The Digital Corpus of Sanskrit*. Berlin.

Oliver Hellwig. 2015. ind. senz–ocr software for hindi, marathi, tamil, and sanskrit.

Hiroshi Ishikawa. 2011. Transformation of general binary mrf minimization to the first-order case. *IEEE transactions on pattern analysis and machine intelligence*, 33(6):1234–1249.

Sittichai Jiampojamarn, Grzegorz Kondrak, and Tarek Sherif. 2007. Applying many-to-many alignments and hidden markov models to letter-to-phoneme conversion. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 372–379, Rochester, New York. Association for Computational Linguistics.

Amrith Krishna, Bishal Santra, Pavankumar Satuluri, Sasi Prasanth Bandaru, Bhumi Faldu, Yajuvendra Singh, and Pawan Goyal. 2016a. Word segmentation in sanskrit using path constrained random walks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 494–504, Osaka, Japan. The COLING 2016 Organizing Committee.

Amrith Krishna, Pavankumar Satuluri, Shubham Sharma, Apurv Kumar, and Pawan Goyal. 2016b. Compound type identification in sanskrit: What roles do the corpus and grammar play? In *Proceedings of the 6th Workshop on South and Southeast Asian Natural Language Processing (WSSANLP2016)*, pages 1–10, Osaka, Japan. The COLING 2016 Organizing Committee.

Praveen Krishnan, Naveen Sankaran, Ajeet Kumar Singh, and CV Jawahar. 2014. Towards a robust ocr system for indic scripts. In *Eleventh IAPR International Workshop on Document Analysis Systems (DAS)*, pages 141–145, Tours, France. IEEE.

Anand Kumar and CV Jawahar. 2007. Content-level annotation of large collection of printed document images. In *Ninth International Conference on Document Analysis and Recognition (ICDAR)*, volume 2, pages 799–803, Parana, Brazil. IEEE.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning (ICML)*, volume 951, pages 282–289, Williamstown, MA, USA.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.

Jey Han Lau, Alexander Clark, and Shalom Lappin. 2015. Unsupervised prediction of acceptability judgements. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1618–1628, Beijing, China. Association for Computational Linguistics.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the*

2015 Conference on Empirical Methods in Natural Language Processing, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Minesh Mathew, Ajeet Kumar Singh, and CV Jawahar. 2016. Multilingual ocr for indic scripts. In *Document Analysis Systems (DAS), 2016 12th IAPR Workshop on*, pages 186–191. IEEE.

Monier Monier-Williams. 1899. A sanskrit-english dictionary.

Umapada Pal, Tetsushi Wakabayashi, and Fumitaka Kimura. 2009. Comparative study of devnagari handwritten character recognition using different feature and classifiers. In *Tenth International Conference on Document Analysis and Recognition (ICDAR)*, pages 1111–1115. IEEE.

Vikas Reddy, Amrith Krishna, Vishnu Dutt Sharma, Prateek Gupta, MR Vineeth, and Pawan Goyal. 2018. Building a word segmenter for sanskrit overnight. In *Eleventh Language Resources and Evaluation Conference (LREC)*, Miyazaki, Japan.

Rohit Saluja, Devaraj Adiga, Parag Chaudhuri, Ganesh Ramakrishnan, and Mark Carman. Error detection and corrections in indic ocr using lstms. In *14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*.

Naveen Sankaran and CV Jawahar. 2012. Recognition of printed devanagari text using blstm neural network. In *21st International Conference on Pattern Recognition (ICPR)*, pages 322–325, Tsukuba Science City, JAPAN. IEEE.

Carsten Schnober, Steffen Eger, Erik-Lân Do Dinh, and Iryna Gurevych. 2016. Still not there? comparing traditional sequence-to-sequence models to encoder-decoder neural networks on monotone string translation tasks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1703–1714.

M. Schuster and K. Nakajima. 2012. Japanese and korean voice search. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152, Kyoto, Japan.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Bikash Shaw, Swapan Kumar Parui, and Malayappan Shridhar. 2008. Offline handwritten devanagari word recognition: A holistic approach based on directional chain code feature and hmm. In *International Conference on Information Technology (ICIT)*, pages 203–208, Bhubaneswar, India. IEEE.

Ajeet Kumar Singh and CV Jawahar. 2015. Can rnns reliably separate script and language at word and line level? In *13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 976–980. IEEE.

Ray Smith. 2007. An overview of the tesseract ocr engine. In *Ninth International Conference on Document Analysis and Recognition, (ICDAR)*, volume 2, pages 629–633. IEEE.

Ray Smith, Daria Antonova, and Dar-Shyang Lee. 2009. Adapting the tesseract open source ocr engine for multilingual ocr. In *Proceedings of the International Workshop on Multilingual OCR*, page 1. ACM.

Raymond W Smith. 1987. *The Extraction and Recognition of Text from Multimedia Document Images*. Ph.D. thesis, University of Bristol.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Adnan Ul-Hasan and Thomas M. Breuel. 2013. Can we build language-independent ocr using lstm networks? In *Proceedings of the 4th International Workshop on Multilingual OCR*, MOCR '13, pages 9:1–9:5, New York, NY, USA. ACM.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, ukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.