Improving the learnability of classifiers for Sanskrit OCR corrections

Devaraja Adiga[1], Rohit Saluja[2], Vaibhav Agrawal[3], Ganesh Ramakrishnan[1], Parag Chaudhuri[1], K. Ramasubramanian[1] and Malhar Kulkarni[1]

[1]Indian Institute of Technology, Bombay, Mumbai, India
[2]IITB-Monash Research Academy, Mumbai, India
[3]Indian Institute of Technology, Kharagpur, India
pdadiga@iitb.ac.in, rohitsaluja@cse.iitb.ac.in, vaibhav@iitkgp.ac.in
{ganesh,paragc}@cse.iitb.ac.in, {ram,malhar}@hss.iitb.ac.in

Abstract

Sanskrit OCR documents have a lot of errors. Correcting those errors using conventional spell checking approaches breaks down due to the limited vocabulary. This is because of high inflections of Sanskrit, where words are dynamically formed by Sandhi rules, Samāsa rules, Taddhita affixes, etc. Therefore, correcting OCR documents require huge efforts. In this paper, we present different machine learning approaches and various ways to improve features for ameliorating the error corrections in Sanskrit OCR documents. We simulated Subanta Prakaraṇam of VaiyākaraṇaSiddhāntaKaumudī for synthesizing off-the-shelf dictionary. Most of the methods we propose can also work for general Sanskrit word corrections.

1 Introduction

Optical character recognition(OCR) is the process of identifying characters in document images for creating editable electronic texts. SanskritOCR by Indsenz, Google OCR and Tesseract are major OCRs available for Sanskrit. Word level error analysis for 6 books printed at various places of India having different fonts scanned with 300 DPI are listed in Table 1. Correcting the errors becomes cumbrous even with the OCR accuracy as high as above 90%, unless complemented by a mechanism for correcting the errors. User feedback based OCR correcting mechanisms can improve through correcting a contiguous text having uniform font. We discuss different approaches for correcting Sanskrit OCR based on available system resources.

Book Name	Publisher Details	Year of Publication	No. of Pages OCRed	WER - IndSenz	WER - Google
Raghuvaṃśam Sanjīvinīsametam	Nirṇaya Sāgara Press, Mumbai	1929	200	19%	35%
Nṛsiṃhapūrvot-taratāpanīyopaniṣat	Ānandāśrama, Pune	1929	160	34%	41%
Siddhānta Śekhara-1	Calcutta University Press	1932	390	38%*	66%
GaṇakaTaraṅgiṇī	Jyotish Prakash Press, Varanasi	1933	150	34%	46%
Siddhānta Śekhara-2	Calcutta University Press	1947	241	55%*	53%
Siddhānta Śiromaṇi	Sampurananda University, Varanasi	1981	596	18%	29%

Table 1: Word Error Rates for Indsenz's SanskritOCR and Google OCR (*After training 5 pages)

Conventional approaches for spell checking uses Levenshtein-Damerrau edit distance to a

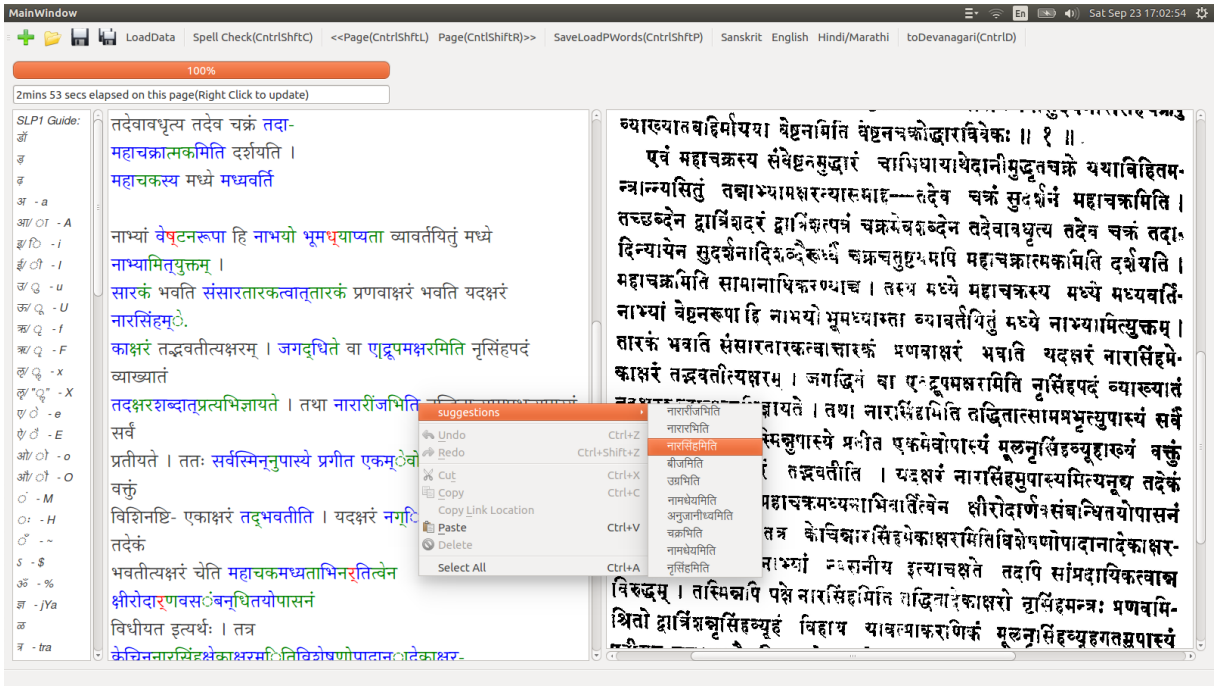

Figure 1: A screen-shot of our framework.

known dictionary and auto-corrects the errors using a language model (Whitelaw et al., 2009). For post-OCR corrections of languages highly rich in inflections, this naive approach results in poor accuracy (Sankaran and Jawahar, 2013). It primarily depends upon lookups into a fixed vocabulary. Such vocabulary for Sanskrit is always incomplete due to words formed dynamically using Sandhi, Samāsa and Taddhita rules.

In recent works, encoder-decoder Recurrent Neural Networks (RNNs) with character-based attention have shown state-of-the-art results in Neural Language Correction (Xie et al., 2016). We (Saluja et al., 2017b) proposed a logistic regression based machine learning framework for correcting Indic OCRs using dual engine OCR. For correcting OCRs across four Indic languages (Sanskrit, Hindi, Kannada and Malayalam) in single engine environment (Saluja et al., 2017a) have succeeded in reaching the state of art using a special type of RNNs, called Long Short Term Memory Networks (LSTM).

OCR Word	Corrected Word	Ground Truth
विशीआआआआर्‌नि	विशीर्णानि	विशीर्णानि
षष्ठ्े ।पनिषत्ि९	षष्ठ्ोपनिषद्ि	षष्ठोपनिषदि
भर्तुर्मुनिराआस्थ्ूएतविष्टरः	भर्तुर्मुनिरास्थ्ितविष्टरः	भर्तुर्मुनिरास्थितविष्टरः
मङ्गलस्तनिस्वनाः	मङ्गलतूर्यनिस्वनाः	मङ्गलतूर्यनिस्वनाः
म ातपक्रं	महाचक्रं	महाचक्रं
नै९वाम्ि०इषच्यते	हैवाभ्ि्ष्ि्च्यते	हैवाभिषिच्यते
भूऊउगव्ूान्य्ुऋउऽCऊ	भगवान्यश्च	भगवान्यश्च

Table 2: Examples of Sanskrit OCR words corrected by our framework.
OCRed data of over 5k Sanskrit document images and 12k in different languages were corrected using our framework - OpenOCRCorrect.

Basic dictionary lookup approach requires less system resources where as Neural language correcting models demands higher system specifications. So we propose and evaluate different

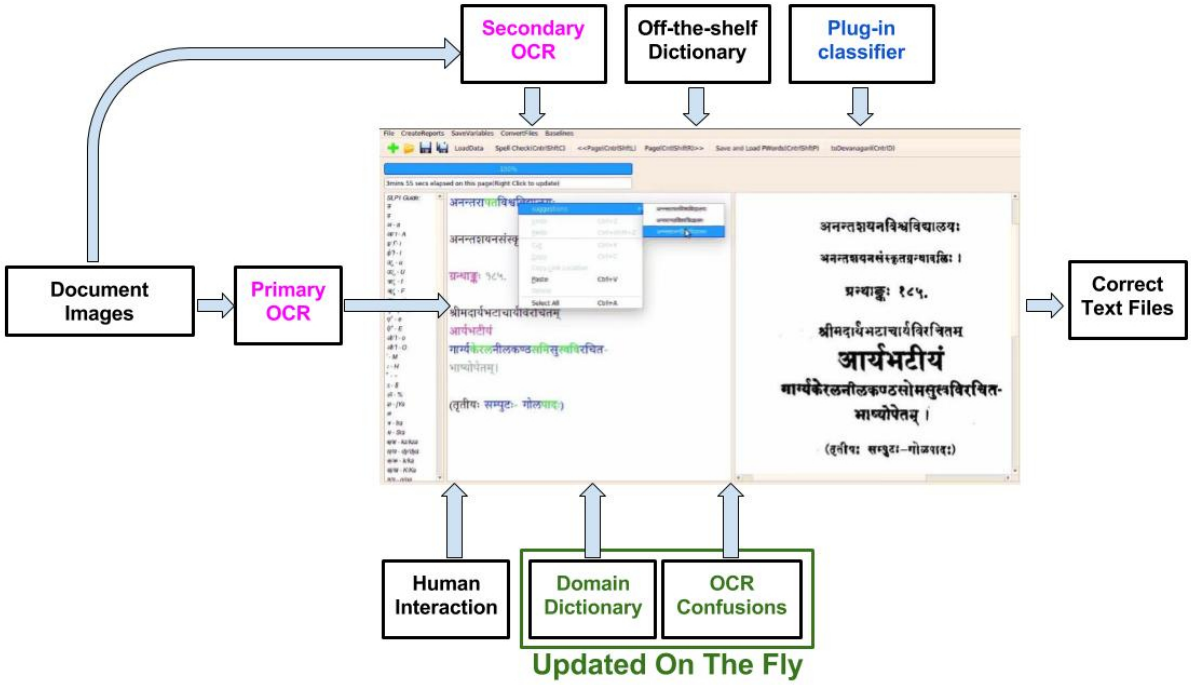

Figure 2: Block diagram of our framework.

models for correcting Sanskrit OCR[1], starting from simple dictionary look up to Neural attention models. For building the vocabulary for Sanskrit, we developed a Subanta-generator. We will be using various other auxiliary sources which we will discuss in the next section. Then in section 4 we discuss the results for various error detecting approaches in detail. Suggestion generation will be explained in the following section. Table 2 shows the OCR errors corrected by our framework and Figure 1 is a screen-shot of our framework. We are using multicolor coding to depict compound words, out-of-vocabulary words, auto-corrected words and correct words.

Our contributions in this paper are i) Suggesting different models for error detection based upon amount of training data (if range is 10k and GPU is not available use Plug-in classfier, for range of 100k with GPU use LSTM and for range of 1000k with GPU use attention models) ii) increasing the learnability of ML classifiers by increasing auxiliary sources and iii) Comparing the different ML based and Deep learning based methods for the task of Error detection. It is important to note that the Plug-in classifier model we have used and LSTM model and results we have shown already exist in the literature for the task of error detection. We have improved the results of Plug-in classifier by introducing more auxiliary sources and synthesized words. Further, we use attention model to compare the results with LSTM based error detection.

2 Auxiliary Sources

Figure 2 depicts the functionality of human-interactive framework for OCR corrections. We will be using various auxiliary sources which are helpful in verifying the correct words and curating the word-level errors. Our system is leveraged by OCR data from different systems, dynamically updated OCR confusions and domain specific vocabulary. We are also using a synthesized off-the-shelf dictionary. These features are used for supervised learning by training a plug-in classifier for achieving better F-score. Erroneous words are corrected using suggestions through human interaction to keep the confidence level high. Later on words having similar errors are auto-corrected. In the following sections we discuss various auxiliary sources used by

[1]The source code of Sanskrit OCR corrector, OpenOCRCorrect is available at https://goo.gl/WqoVi2

the framework.

2.1 OCR documents from different systems

Since different OCR systems are using different models they are likely to make different kinds of errors and are likely to be correct on the OCR words that they agree upon. This observation is especially leveraged by ensemble-based ML approach (Polikar, 2006). Therefore OCR documents from different systems can become a powerful auxiliary source.

2.2 Off-the shelf dictionary

Since the vocabulary is incomplete for Sanskrit due to rich inflections, we developed a Subanta generator for synthesizing noun variants. Among the different declension generators, (Patel and Katuri, 2015a) is an open source Subanta generator for Sanskrit. We developed a new Subanta generator for the following reasons

- For ease of integration into the OCR framework
- For overcoming the errors produced by existing Subanta generator. Examples from (Patel and Katuri, 2015b) -
  - **प्रथमा एकवचनम्** for words ending with **ऋ**.
  - **द्वितीया द्विवचनम्** for many of the Sarvanāmas.
  - Declensions for words ending with **वसु** affix are wrong in case of **भसंज्ञा**.
- To have the provision for future enhancements

Aṣṭādhyāyī rules corresponding to Subanta Prakaraṇam and required Sandhi rules are coded in accordance with the rules explanations as given in (Dīkṣita et al., 2006). For resolving the conflicts we chose the order of applicability of rules as per the Paribhāṣā - **परनित्यान्तरङ्गापवादानामुत्तरोत्तरं बलीयः**. Context dependencies of many rules are resolved by collecting the context informations. For example, for the rule **एकाचो बशो भष् झषन्तस्य स्ध्वोः** (अ.8-2-37), the roots collected are **गाध्, गुध्, गृध्, दघ्, दध्, दभ्, द्राघ्, बध्, बीभ्, बुध्**. And also the roots **गाह्, गुह्, गृह्, ग्रह्, ग्लह्, दह्, दिह्, दुह्, दृह्, द्राह्, द्रुह्, बाह्, बृह्** are considered after applying '**दादेर्धातोर्घः**' (अ.8-2-32) or '**हो ढः**' (अ.8-2-31).An example word where this rule is applied - **कामधुक् (प्रातिपदिकम् - कामदुह्)**. For the rules **नाभ्यस्ताच्छतुः** (अ.7-1-78), **आच्छीनद्योर्नुम्** (अ.7-1-80) and **शप्श्यनोर्नित्यम्** (अ.7-1-81), we grouped the participles of roots belonging to different conjugations accordingly. Similar way we tried to completely/partially solve the context dependencies of many rules.

We have processed XML file of Monier-Williams Sanskrit Dictionary available in the Cologne Digital Sanskrit Dictionary collections, Institue for Indology and Tamilistics, University of Cologne (http://www.sanskrit-lexicon.uni-koeln.de/download.html) and extracted more than 1.8 lakh words with the gender information from the XML file. Vibhakti variants for these words are generated using the Subanta generator and around 3.2million unique words are generated. We also used the verbs which are listed in the **क्रियारूपनिष्पादिका** (Verb-forms-Generator) of ILTP-DC, which are around 3 lakh unique words. These 3.5 million words are used as off-the-shelf dictionary for the OCR corrector.

2.3 Domain specific vocabulary

In Sanskrit literature frequency of commonly used words changes from one Śāstra to another. So the domain specific vocabulary is most powerful auxiliary resource which will fill the words not found in off-the-shelf dictionary. Domain specific vocabulary is created by extracting unique strings from the various books available in Göttingen Register of Electronic Texts in Indian Languages (GRETIL,). This auxiliary source is also dynamically updated as the user corrects the document, which helps in correcting rest of the document.

2.4 Sandhi Rules

Due to Sandhi rules and Samāsa, words can change dynamically in Sanskrit documents. We are using basic Sandhi rules to find the subwords of a compound word and to match with the words from the vocabulary for detecting its correctness. A greedy approach is used for this splitting with minimum set of words of maximum length and minimum edit distance as the criteria. For examample, the OCR word जागरितावस्थायाभेवावस्थात्रयमुक्तं will be split into जागरित, अवस्थायाभ् (this word is matched with अवस्थायाम्), एव, अवस्थात्रयम् and उक्तं. This helps in detecting out-of-vocabulary words and generating suggestions for them.

2.5 Document and OCR specific n-gram confusions

Since different OCR systems use different preprocessing techniques, different classifier models, error confusions for a word varies from one OCR engine to another (Abdulkader and Casey, 2009). Thus, the OCR specific confusions can be helpful in deciding whether the part of the erroneous word should be changed or not and also in deciding the tie while changing the part of the erroneous word. For example, while changing the erroneous word निवन्ध्यः, if the dictionary lookup suggests निबन्धः and निरन्धः as nearest possible words, having higher n-gram confusion to व->ब biases the selection towrds निबन्धः.

3 Methodologies Followed

3.1 Learning by Optimizing Performance Measures though Plug-in Approach

We rephrase our basic problem of error detection as that of continuously evolving a classifier that labels the OCR of a word as correct or incorrect. The classifier should be trained to optimize a performance measure that is not necessarily the conventional likelihood function or sum of squares error. An example performance measure to be maximized and that is coherent with our needs of maximizing recall (coverage) in detecting erroneous words while also being precise in this detection is the $F-$score, which, unfortunately does not decompose over the training examples and can be hard to optimize. We adapt a plug-in approach (Narasimhan et al., 2014) to train our binary classifier over such non-decomposable objectives while also being efficient for incremental re-training.

Consider a simple binary classification problem where the task is to assign every data point $\mathbf{x} \in \mathcal{X}$, a binary label $y \in \pm 1$. Plug-in classifiers achieve this by first learning to predict *Class Probability Estimate* (CPE) scores. A function $g : \mathcal{X} \rightarrow [0, 1]$ is learned such that $g(\mathbf{x}) \approx \Pr(y = 1)$. Various tools such as logistic regression may be used to learn this CPE model g. The final classifier is of the form $sign(g(\mathbf{x}) - \eta)$ where η is a threshold that is tuned to maximize the performance measure being considered, e.g. F-measure, G-mean etc.

In (Saluja et al., 2017b), various features based on dictionary n-grams and language rules have been used in Sanskrit, Hindi and Marathi. Our major work in this paper is to improve features for such a classifier and verify their effect in three different domains in Sanskrit. We use train:val:test ratio as 48:12:40 for all our experiments that use Plugin classifier since we wanted to explore the possiblity of using the classifier to correct last 40% of book, once initial 60% of the book is corrected.

3.2 LSTM with fixed delay

The basic RNN (Recurrent Neural Network) can be represented by Equations 1 and 2.

$$h_t = g(W_{hh}h_{t-1} + W_{hx}x_t + b_h) \tag{1}$$

$$y_t = W_{yh}h_t \tag{2}$$

g can be sigmoid($\sigma(x_i) = \frac{exp(x_i)}{\sum_j [exp(x_j)]}$) or tanh ($tanh(x) = 2\sigma(2x) - 1$) or Rectified Linear Unit (ReLU) ($f(x) = \max(0, x)$) (Talathi and Vartak., 2014). The matrices W_{hx} and W_{yh} connect the input to the hidden layer and hidden layer to output respectively. These matrices are common

for instance in the sequence. The matrix W_{hh} is the feedback from past input and is responsible for remembrance and forgetfulness of the past sequence based on context.

Equation 2 at each time t can be unfolded back in time, to time t = 1 for the 1^st character of the word sequence, using Equation 1 and the network can be trained using back-propagation through time (BPTT) (Schuster and Paliwal., 1997).

Since we have taken care to ensure equal byte length per letter with ASCII transliteration scheme, for the loss function we used negative log-likelihood of Log SoftMax (multi-class) function. The Log SoftMax function is given in 3, where y_{t_i} is the value at i^{th} index of output vector y_t.

$$f(y_{t_i}) = log(\frac{exp(y_{t_i})}{\sum_j [exp(y_{t_j})]}) \tag{3}$$

The equations are similar for the LSTM with each unit as a memory unit, instead of a neuron. Such memory unit remembers, forgets, and transfers cell state to the output(or next state) based on input history. The cell state at time t is given by equation 4 where the forget gate f_t and the input gate i_t fire according to equations 5 and 6 respectively.

$$c_t = f_t c_{t-1} + i_t g_1(W_{hc} h_{t-1} + W_{xc} x_t + b_c) \tag{4}$$

$$f_t = g_2(W_{xf} x_t + W_{hf} h_{t-1} + W_{cf} c_{t-1} + b_f) \tag{5}$$

$$i_t = g_2(W_{xi} x_t + W_{hi} h_{t-1} + W_{ci} c_{t-1} + b_i) \tag{6}$$

The data is selectively transferred from the cell to hidden state h_t according to equation 7 where the selection is done by the firing of output gate o_t as per equation 8.

$$h_t = o_t g_3(c_t) \tag{7}$$

g_1 and g_3 are generally tanh and g_2 is generally sigmoid.

$$o_t = g_2(W_{xo} x_t + W_{ho} h_{t-1} + W_{co} c_{t-1} + b_o) \tag{8}$$

(Saluja et al., 2017a) uses 512 X 2 LSTM (Long Short-Term Memory based neural network) with the fixed delay (between input sequence and output sequence) trained and tested on characters from 86k OCR word correction pairs with train:val:test split as 64:16:20. The model when trained on large data is able to learn OCR specific error patterns as well as language model. The model abstains from changing the correct word. Thus, for error detection, the word changed by such model is marked as incorrect whereas the word unchanged by the model is marked as correct. Such model over-fits the OCR system and domain. It works well with dataset range of hundred thousand OCR word correction pairs.

3.3 Attention Model

Here, we use the model with more number of layers than LSTM based model discussed in previous section. Attention models are the models with a separate encoder as well as a decoder as compared to LSTM based model wherin the same layers encode as well as decode a sequence. Attention models contain RNN layers as encoder that can take characters from OCR words, and similar RNN based decoders decode the encoder's output to correct word when trained with large amount of data. The attention layers, which are applied on encoder's ourput to help the decoder, learn to give attention to different context around the character being corrected based on the input. We train and test such model with the 86k OCR word correction pairs used in (Saluja et al., 2017a). We use open source library OpenNMT (`http://opennmt.net/`) for this purpose with the default model that includes 500 X 2 LSTM encoder as well as 500 X 2 LSTM decoder. Such model is able to learn dataset of order of millions of OCR word correction pairs as per our experiences for French and English in ICDAR Post-OCR Competition 2017.

Here again, we mark words changed by the model as incorrect for error detection and the words that remained unchanged as correct. As we will see later, even when trained on 86k pairs, such model is able to perform close to LSTM based model for error detection task.

4 Error Detection Methods and Results

4.1 Unsupervised approach

#	Approach	TP	FP	TN	FN	Precision	Recall	F-Score
1.	General Dict. Lookup	89.18	40.12	59.87	10.82	29.75	89.18	44.61
2.	Sandhi Rules	54.34	13.23	86.77	45.66	43.89	54.34	48.56
3.	Secondary OCR Lookup	90.68	23.59	76.40	9.31	42.79	90.68	58.14

Table 3: Error Detection Results with unsupervised methods. Using Sandhi rules while dictionary lookup increase true detections(TN) but increase false detections(FN) as well which is balanced by secondary OCR lookup.

We applied various methods for detecting errors in the OCR text. To start with, we used the book named "Āryabhaṭīyabhāṣya of Nīlakaṇṭha III Golapāda(AnantaśayanaSaṃskṛtaGranthāvaliḥ, 1957)" for which we had the OCR text (OCRed from indsenz) and the ground truth data available.

Using unsupervised methods, commonly used dictionary lookup based approach gave poor F-Scores due to lot of correct words marked as errors, i.e. lower True Negative percentage as shown in Table 3. Marking the words that are formed by applying Sandhi rules on dictionary lexicons as correct increased detection of correct words(True Negatives) but not the detection of errors (True Positives) as compared to previous approach. For this book, lookup into OCR output of other engine (Google Docs) for the same document images improved the F-Score to a decent value.

4.2 Single Engine Environment

#	Approach	TP	FP	TN	FN	Prec.	Recall	F-Score
1.	Classifier with ngrams frequency + word lookup in General Dict. as features	73.38	22.86	77.13	26.61	38.88	73.38	50.83
2.	Classifier with ngrams frequency + word lookup in Synthesised Dict.(superset of gen. dict.) as features	74.06	21.02	78.98	25.94	41.14	74.06	52.89
3.	Classifier with ngrams frequency + word lookup in Synthesised Dict. as well as Domain Dict. as features	66.37	13.08	86.92	33.63	50.38	66.37	57.28
4.	Classifier with features in row 3 + no. of Sandhi components in OCR word as features	68.50	13.53	86.47	31.50	50.10	68.50	57.87

Table 4: ML Classifier's Error Detection Results in Single Engine Environment. Here we achieved the F-score close to that of Secondary OCR lookup using Unsupervised approach

For supervised learning using the plug-in classifier as explained in section 3.1, we are splitting the data with train:val:test ratio as 48:12:40, we train the plug-in classifier with various features. We are able to improve the row 1 results in Table 3 by including frequency of ngrams (upto 8)

in general dictionary as features. We also include the binary feature based on lookup in general dictionary. The results are shown in first row of Table 4.

We further include more words in the dictionary by synthesizing nouns and collecting the verbs as explained in 2.2. This help us to achieve the results shown in row 2 of Table 4.

Adding frequencies of n-grams from OCR word as features from domain dictionary generated as explained in 2.3 along with synthesized dictionary improved the results as shown in row 3 of Table 4.

For improving the results further as shown in row 3 of Table 4, we used three splitting based features. i) Split the OCR words using commonly used Sandhi rules and used the no. of lexicon components obtained from the general dictionary as features. ii) We also used no. of lexicon components obtained by splitting the OCR word as lexicons of domain dictionary (for Jyotiṣa) as a feature. Herein, the no. of characters from unknown sub-strings in the OCR word are added to the feature. iii) The product of features obtained in (i) and (ii) is also used as the feature. We normalized all these features about the mean and standard deviation of training data.

The results are shown in row 4 of Table 4. It is important to note that here in single engine environment we are able to reach closer to the dual engine environment based Secondary OCR Lookup approach given in the row 3 of Table 3.

4.3 Multi Engine Environment

#	Approach	TP	FP	TN	FN	Prec.	Recall	F-Score
1.	Classifier with features in table 4 row 2 along with dual engine agreement*	85.13	17.84	82.16	14.87	48.62	85.13	61.89
2.	Classifier with features in table 4 row 3 along with dual engine agreement	78.04	13.67	86.33	21.96	53.11	78.04	63.20
3.	Classifier with features in table 4 row 4 along with dual engine agreement	83.49	15.26	84.74	16.51	52.25	83.49	64.28
4.	Classifier with features in table 4 row 4 along with triple engine agreements	83.43	14.95	85.04	16.56	52.74	83.43	64.63

Table 5: ML Classifier's Error Detection Results in Multi Engine Environments. (*state of the art (Saluja et al., 2017b)). Here TP is significantly increased when compared to single engine environment.

We further include the dual engine OCR agreement as feature in addition to the features used in previous sections and achieve the results obtained in Table 5. Here we have used Indsenz as primary OCR engine and Google docs as secondary OCR engine.

We improve the results further by using the feature of dual OCR agreement between Indsenz and Tesseract in addition to previous features to obtain the results shown in row 4 of Table 5.

We present the results of Plug-in Classifier trained and tested on the dataset of books with different domains in Table 6 for proving its consistency over various domains. Row 1 in this table shows the baseline for the book 'Nṛsiṃhapūrvottaratāpanīyopaniṣat' (ĀnandāśramaSaṃskṛta-Granthāvaliḥ, 1929) and row 2 shows the results achieved using all the features (obtained using triple engine environment, off-the-shelf dictionary, domain vocabulary and ngram frequency from general, synthesized and domain vocabularies). It is important to note that the TP (Errors detected as errors) is high for the baseline in this case as compared to TP for baseline in other domains. However, TN (Correct words detected as correct) for the dictionary lookup baselines are however close to each other for all domains as shown in row 1 of Table 3 and row

#	Approach	TP	FP	TN	FN	Prec.	Recall	F-Score
1.	Vedānta gen. dict. lookup baseline	85.52	34.35	65.65	14.48	49.71	85.52	62.87
2.	Vedānta Plug-in Classifier	79.95	9.80	90.20	20.05	77.95	79.95	78.94
3.	Sāhitya gen. dict. lookup baseline	64.24	35.36	64.64	35.76	32.86	64.24	43.49
4.	Sāhitya Plug-in Classifier	87.88	13.37	86.62	12.12	66.52	87.88	75.72

Table 6: ML Classifier's Error Detection Results for other domains. Above results shows the generality of the model for different domains of Sanskrit literature.

1 and row 3 of Table 6. The reason for high TN could be less ambiguity (as compared to other domains) in incorrect words, since TP (unlike TN) does not depend on the presence of correct OCR words in dictionary. Hence we are getting F-score as high as 62.87 for the baseline in this case. We also evaluated the system for Sāhitya domain. For this we have used the book 'Raghuvaṃśam Sanjīvinīsametam' (Nirṇaya Sāgara Press, 1929, 1-9 Sarga) and row 3 in table 6 shows the baseline, whereas row 4 shows the results obtained using our framework.

4.4 Deep Neural Network based approaches

#	Approach	TP	FP	TN	FN	Prec.	Recall	F-Score
1.	LSTM with fixed delay*	92.64	5.45	94.54	7.36	94.84	92.64	93.72
2.	Char. level Attention model	81.53	7.74	92.26	18.47	91.92	81.53	86.41

Table 7: Neural Network's Error Detection Results. (*state of the art (Saluja et al., 2017a))

Here, in Table 7, we present the results for the approaches described in Sections 3.2 and 3.3 for 86k pairs used in (Saluja et al., 2017a) with 64:16:20 as train:val:test split. The first row shows the Sanskrit results from (Saluja et al., 2017a). The second row present the results for character level attention model. For attention model, we use characters from OCR word and its preceding OCR word (as context) at input and characters from correct word at output. We tried other contexts at input as well. Using the context of characters from one word gave optimized F-Score.

F-scores show that, using these approaches we can outperform all other ML techniques, but requires large amount of training data for generic adaptations. Since, these models learn error patterns and language based on dataset, if the test data differs (in terms of OCR confusions/system and/or domain from training data), we can make use of approaches mentioned in the previous sections. Since Plug-in-classifier is uses general auxiliary sources, we recommend to use it for practical purposes.

5 Suggestion Generation

The results for various ways of exploiting auxiliary sources, to generate appropriate suggestions, are given in (Saluja et al., 2017b) for "Āryabhaṭīyabhāṣya of Nīlakaṇṭha III Golapāda(1957)".

Here, in Table 8, we show the improvement in results due to adaptations of domain dictionary and OCR Confusions on-the-fly for "Āryabhaṭīyabhāṣya of Nīlakaṇṭha III Kālakriyāpāda(AnantaśayanaSaṃskṛtaGranthāvaliḥ, 1931)".

We synthetically generated word images for the words in Sanskrit dictionaries, and OCRed them using ind.senz (Indsenz,), and extracted around 0.5 million erroneous-correct word pairs. We used the longest common subsequence algorithm (Hirschberg, 1977) for generating around 0.78 million OCR character confusions. The row 1 of Table 8 shows the total percentage of correct suggestions obtained using various auxiliary sources with i) words common to dual

Sources Included	Percentage of Correct Suggestions
Domain words with dual OCR agreement + Synthesized Confusions	36.26
Prev. + adapting Domain Words/Page	36.38
Prev. + adapting Confusions/Page	37.14
Prev. - Synthesized + Real Confusions	39.40

Table 8: Improvement in Suggestions with Adaptive sources for "Āryabhaṭīyabhāṣya of Nīlakaṇṭha III Kālakriyāpāda(AnantaśayanaSaṃskṛtaGranthāvaliḥ, 1931)".

OCR systems as Domain Vocabulary throughout the document and ii) obtained synthesized confusions. As shown in row 2, we further improved the quality of suggestions by uploading the corrected domain words on-the-fly after the user corrects the page. Adapting the confusions on-the-fly page by page further improved results as shown in row 3. Using real confusions from the primary OCR text and ground truth from other books further helped in improving results as shown in row 4 of Table 8.

6 Conclusions

In this paper we demonstrate different ML approaches for Sanskrit OCR corrections. Our framework leverages synthesized dictionary, n-gram error confusions and domain vocabularies. Error confusions and domain specific vocabularies grow on-the-fly with user corrections. We have presented a multi-engine environment which is useful in detecting potential errors. Using various auxiliary sources along with plug-in classifier we succeed in achieving F-Scores better than (Saluja et al., 2017b). LSTM with fixed delay is outperforming other approaches. Deep neural network based approaches, however, require higher level resources like GPU and large amount of training data. Our system is able to generate correct suggestions for the errors having edit distance as high as 15. As shown in (Saluja et al., 2017b), our GUI is able to reduce the overall cognitive load of the user by providing adequate color coding, generating suggestions and auto-correcting similar erroneous words. For Sandhi splitting using Saṃsādhanī's **सन्धिविच्छेदिका**[2] can be a better option than the greedy approach which can be considered for future enhancement to the framework.

References

Ahmad Abdulkader and Matthew R. Casey. 2009. Low cost correction of ocr errors using learning in a multi-engine environment. In *Proceedings of the 10th international conference on document analysis and recognition*.

Bhaṭṭojī Dīkṣita, Vāsudeva Dīkṣita, and Jñānendra Sarasvatī. 2006. *Vaiyākaraṇasiddhāntakaumudī with the commentary Bālamanoramā and Tattvabodhinī*. Motilal Banarasidas.

GRETIL. -Göttingen Register of Electronic Texts in Indian Languages. http://gretil.sub.uni-goettingen.de/gretil.htm.

Daniel S Hirschberg. 1977. Algorithms for the longest common subsequence problem. *Journal of the ACM (JACM)*, 24(4):664–675.

Indsenz. SanskritOCR. *http://www.indsenz.com/*. Last accessed on 09/15/2017.

Harikrishna Narasimhan, Rohit Vaish, and Shivani Agarwal. 2014. On the statistical consistency of plug-in classifiers for non-decomposable performance measures. In *Proceedings of NIPS*.

Dhaval Patel and Shivakumari Katuri. 2015a. Prakriyāpradarśinī - an open source subanta generator. In *16th World Sanskrit Conference*.

[2]**संसाधनी - सन्धिविच्छेदिका** http://scl.samsaadhanii.in/ by Amba Kulkarni

Dhaval Patel and Sivakumari Katuri. 2015b. Subanta generator. *http://www.sanskritworld.in/sanskrittool/SanskritVerb/subanta.html*. Last accessed on 09/30/2017.

R. Polikar. 2006. Ensemble based systems in decision making. In *IEEE Circuits and Systems Magazine*.

Rohit Saluja, Devaraj Adiga, Parag Chaudhuri, Ganesh Ramakrishnan, and Mark Carman. 2017a. Error detection and corrections in Indic OCR using LSTMs. *International Conference on Document Analysis and Recognition (ICDAR)*.

Rohit Saluja, Devaraj Adiga, Parag Chaudhuri, Ganesh Ramakrishnan, and Mark Carman. 2017b. A framework for document specific error detection and corrections in Indic OCR. *1st International Workshop on Open Services and Tools for Document Analysis (ICDAR- OST)*.

Naveen Sankaran and C.V. Jawahar. 2013. Error detection in highly inflectional languages. In *Proceedings of 12th International Conference on Document Analysis and Recognition*, pages 1135–1139. IEEE.

Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. In *IEEE Transactions on Signal Processing*.

Sachin S. Talathi and Aniket Vartak. 2014. Improving performance of recurrent neural network with relu nonlinearity. In *In the International Conference on Learning Representations workshop track*.

Casey Whitelaw, Ben Hutchinson, Grace Y Chung, and Gerard Ellis. 2009. Using the web for language independent spellchecking and autocorrection. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing: Volume 2*, pages 890–899. Association for Computational Linguistics.

Ziang Xie, Anand Avati, Naveen Arivazhagan, Dan Jurafsky, and Andrew Y. Ng. 2016. Neural language correction with character-based attention. *arXiv preprint arXiv:1603.09727*.