Error Detection and Corrections in Indic OCR using LSTMs

Rohit Saluja
IITB-Monash Research Academy
Mumbai, India
rohitsaluja@cse.iitb.ac.in

Devaraj Adiga
IIT Bombay
Mumbai, India
pdadiga@iitb.ac.in

Parag Chaudhuri
IIT Bombay
Mumbai, India
paragc@cse.iitb.ac.in

Ganesh Ramakrishnan
IIT Bombay
Mumbai, India
ganesh@cse.iitb.ac.in

Mark Carman
Monash University
Victoria, Australia
mark.carman@monash.edu

***Abstract*—Conventional approaches to spell checking suggest spelling corrections using proximity-based matches to a known vocabulary. For highly inflectional Indian languages, any off-the-shelf vocabulary is significantly incomplete, since a large fraction of words in Indic documents are generated using word conjoining rules. Therefore, a tremendous manual effort is needed in spell-correcting words in Indic OCR documents. Moreover, in a spell checking system, a vocabulary may suggest multiple alternatives to the incorrect word. The ranking of these corrective suggestions is improved using language models. Owing to corpus resource scarcity, however, Indian languages lack reliable language models. Thus, learning the character (or n-gram) confusions or error patterns of the OCR system can be helpful in correcting the Out of Vocabulary (OOV) words in OCR documents. We adopt a Long Short-Term Memory (LSTM) based character level language model with a fixed delay for discriminative language modeling in the context of OCR errors for jointly addressing the problems of error detection and correction in Indic OCR. For words that need not be corrected in the OCR output, our model simply abstains from suggesting any changes. We present extensive results to validate the performance of our model on four Indian languages with different inflectional complexities. We achieve F-Scores above 92.4% and decreases in Word Error Rates (WER) of at least 26.7% across the four languages.**

I. Introduction

Optical Character Recognition (OCR) is the process of converting document images to editable electronic format [1]. Manual correction of OCR documents is very cumbersome. An OCR system even with accuracies as high as 90% is not sufficiently useful unless complemented by a partially automated mechanism for error detection and correction. Thus, error detection can be considered a very important step in post processing OCR words. On large contiguous texts with nearly uniform font and language characteristics, it may be further desirable that the error detection and correction system be able to continuously improve itself by incorporating user feedback. The importance of post-OCR text correction has been emphasized in literature [2], [3] and is further highlighted by the introduction of a recent competition for comparing systems for such corrections in English and French documents [4].

Conventional approaches to spell-checking and correction yield inadequate accuracies in the context of post-OCR corrections for languages that are rich in inflections [5]. These primarily depend upon lookups into fixed vocabularies. Such vocabularies are largely incomplete for languages in which words are dynamically formed using word conjoining rules.

OCR Word	LSTM output/ Correct OOV Word
एवमसक्तात्करणेऽवनिघ्नीः	एवमसकृत्करणेऽवनिसूनोः
കല്ലാന്റെടുത്തുകൊണ്ടിരിക്കുന്ന	ക്ലാസ്സെടുത്തുകൊണ്ടിരിക്കുന്ന
ಗಜಸೆನಿವನು	ಗಜಸೈನ್ಯವನ್ನು
जसदपाऊ	जसदयाल

Fig. 1. Examples of OCR words corrected by LSTM in four Indic languages. Here, the correct words are all OOV words. Mistakes are marked in red.

A crude example of word conjoining rules from English involves placing together the words “every” and “one” to form the word “everyone”. A simple example from Sanskrit is of combining “viṣaya”, “pañcaka” and “rahitam” together to form “viṣayapañcakarahitam”. The second column of Figure 1 includes conjoint words for 4 Indian languages generated using complex conjoining rules.

In recent work, Recurrent Neural Networks (RNNs) have been found to be effective in learning character level language models [6]. Such models make no local independence assumptions on the language text, unlike Hidden Markov Models (HMM). Character-based attention RNNs have also shown state-of-the-art results in Neural Language Correction [7]. In particular, a special type of RNNs, called Long Short Term Memory Networks (LSTMs), remember larger contexts and are therefore best suited for languages rich in OOV words. We delay the output in LSTM model to take care of n-gram character confusions and succeeding contexts. In this paper, we present a model based on LSTM with a fixed delay that can learn, detect and correct OCR errors in addition to learning the language model. Examples of errors detected and corrected by our model are shown in Figure 1. In the process of manually correcting OCR documents, we frequently observe that the knowledge of error patterns is helpful in error detection and correction. The OCR system tends to get confused between letters with similar images. To correct such errors, we need not refer to the original word image, since the information can be inferred from the error patterns and the context in OCR words. This observation is supported by our results.

We present the problem scope and error analysis in Section III, wherein we also describe our dataset. Subsequently, in Section IV, we present our method of using an LSTM with a fixed delay for OCR correction in Indian Languages. In

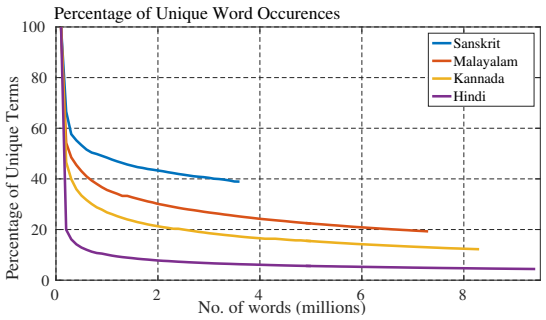

Fig. 2. Unique word coverage between Sanskrit, Malayalam, Kannada and Hindi (Top to bottom). Higher curve implies more OOV words in the language.

Section V, we elaborate on our experiments, including various ways in which we exploit our model in different contexts. The results are summarized in Section VI, followed by conclusions and directions for future work in Section VII.

II. Related Work

Dictionary-based approaches to corrections have been explored in many works in the past [8], [9]. These methods typically perform poorly in the absence of complete vocabularies. Some of the works also concentrate on making the dictionary based approach more efficient [10]. A trigram-based noisy-channel model has been used to detect word errors [11] and make context-sensitive corrections. Part-of-speech trigrams have also been combined with Bayesian methods for context-sensitive spelling correction [12]. More complex methods [13] employ a combination of models such as a shape classifier model, a word n-gram model & a binary n-gram dictionary model to detect errors. These methods, however, do not perform well for the inflectionally rich Indian languages.

OCR error detection for highly inflectional Indian languages faces challenges such as large unique word lists, lack of linguistic resources & lack of reliable language models [5]. This work shows that the Support Vector Machine (SVM) classifier performs better than a conventional technique of dictionary lookup (combined with n-grams) for Malayalam and Telugu. More recently, RNNs and Gaussian Mixture Models have been used to detect the erroneous OCR words in Hindi, Gujarati, Malayalam, & Telugu [14], and these models outperform the SVM model [5] on precision, recall & F-Score.

There have been earlier efforts on post-processing the OCR output of individual Indian languages. A statistical sub-character language model, that backs-off a multi-stage graph module, was used in the post-processing scheme for Malayalam OCR [15]. A shape-based system for post processing Gurmukhi OCR output has also been investigated [16]. The technique of morphological parsing has been applied for character-level error correction in Bangla OCR text [17]. Finally, difficulties in developing spell-checkers for Hindi, Bengali & English involving real-word errors (RWE) and non-word errors (NWE) were discussed in Choudhury et al. [18].

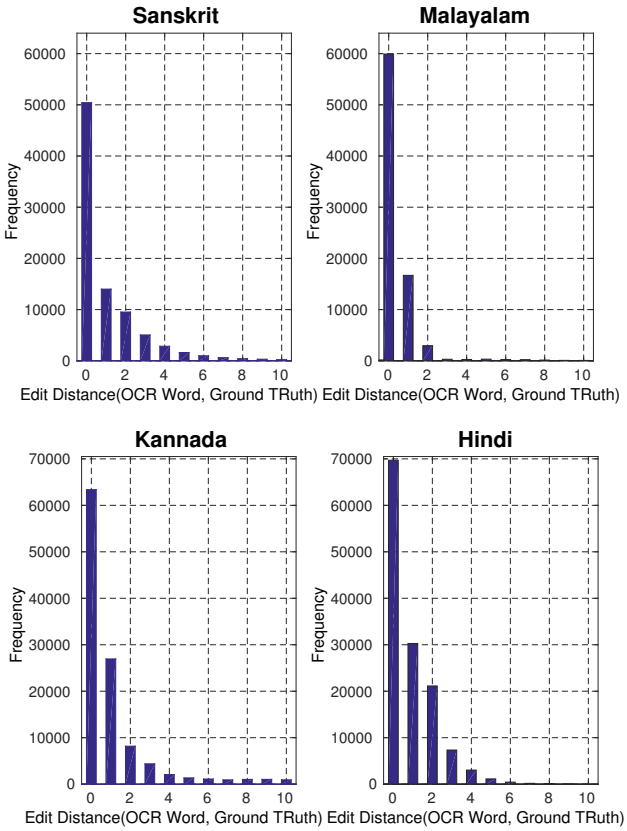

Fig. 3. Histogram of edit distance between OCR word and corresponding ground truth word pairs. As shown, most of the erroneous OCR words (with edit distance ≥ 1) exhibit confusion on one character and the frequency of erroneous words exponentially decreases with edit distance.

III. Problem scope, Data Description & analysis

Some vocabulary characteristics (such as dynamism and size) can be analyzed through the graphs of unique words versus corpus length. This has been studied for Malayalam and Telugu [5]. In Figure 2, we present a similar analysis for Sanskrit, Malayalam, Kannada and Hindi. As shown, the vocabulary is most dynamic/incomplete in Sanskrit, followed by Malayalam, Kannada, and Hindi.

A glimpse of OCR errors can be obtained using the histogram of Levenshtein-Damerau edit distance between pairs of OCR words and corresponding ground truth word. We chose to work using edit distance, instead of hamming distance, since a significant fraction of OCR errors consists of confusion between letters that look similar.

We used Google's free OCR-service [19] to scan $86k$ Sanskrit, $81k$ Malayalam and $118k$ Kannada words from different documents. We carefully corrected the OCR words to form the ground truth. For Hindi, we used $67k$ word pairs from a state-of-the-art work [14]. Since more than 96% of the words were incorrect in the original dataset for Hindi, we balanced the dataset by also including "Ground Truth, Ground Truth word" pairs in addition to the "OCR word, Ground Truth word" pairs. This increased the number of word pairs in Hindi to $134k$. We aligned the word pairs using the recursive

Text Alignment Tool (RETA) [20]. To tackle the problem of variable byte length per character in Indic scripts, we used ASCII transliteration schemes such as SLP1 (Sanskrit Library Phonetic Basic encoding scheme) for all our experiments.

The histograms for Sanskrit, Malayalam, Kannada, and Hindi word pairs are shown in Figure 3. The frequency at 0 edit distance represents the number of words correctly detected by the OCR system. It is important to note that the frequency of erroneous OCR words, *i.e.*, words with edit distance ≥ 1, is maximum for the words that are a unit distance away from the ground truth. This frequency decreases exponentially with edit distance irrespective of the language and the OCR system. A good OCR system would tend to make fewer mistakes at higher edit distances, and thus such a histogram would decay faster as compared to a poor quality OCR system. Interestingly, such a histogram provides intuition regarding the appropriate amount of delay in our LSTM model (7 for Sanskrit & Kannada and 5 for Malayalam & Hindi).

IV. LSTM Model for post-OCR Error Detection and Correction

An LSTM can be used to predict characters that appear in a word based on a preceding sequence of characters. Character-based approaches have not (yet) attained state-of-the-art performance on language modeling tasks [7], but such an approach can be useful for correcting OCR errors since the OCR output is partially correct and most errors follow some known confusion patterns based on images of characters that look similar. An erroneous character in a word can be more robustly detected and corrected if we look at the sequence of characters that appear before and after the character. This is modeled by an LSTM with a fixed delay, wherein the delay allows the succeeding sequence of characters to also be used for learning, unlike a simple LSTM where only the preceding sequence is considered. The length of the future sequence used is equal to the delay. Another reason for including the delay is to allow for character contractions, whereby multiple characters are replaced by a single character.

We aligned word pairs from OCR documents and the corresponding ground truth documents. We used one-hot-encoded characters (see the input layer of Figure 4) from a word in the OCR document as input and characters, from the corresponding aligned word in the ground truth documents as output, for training and testing our model. Our results show that an LSTM model, with a fixed delay at the output, trained in this manner is capable of word-level error detection and correction.

In Figure 4, we illustrate an LSTM with 2 units of delay, with one hidden layer of 3 units, unfolded for 8 units of time. We represent the delay by making use of null-character ($) symbols. The OCR word at the input is delayed with a buffer of 1 unit and the corresponding ground truth word at the output is delayed by 3 units of time to account for 2 units of net delay. We illustrate an example encoding for the Devanagari letters of Figure 4 in Table I. We assume that the character vocabulary size and the maximum word length are both 8.

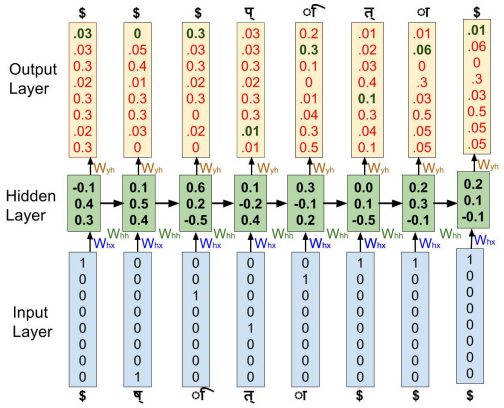

Fig. 4. An LSTM model with 2 units of delay (appears as character $), having 1 hidden layer of 3 units, unfolded for 8-time units.

Devanagari Letter	Corresponding SLP1 char.	One Hot Vector
$	$	10000000
ा	A	01000000
ि	i	00100000
त्, ्त	t	00001000
प्, ्प	p	00000010
्ब, ब्	z	00000001

TABLE I
One Hot Vector and corresponding SLP1 character for Devanagari letters used in Fig. 4.

Training the model should produce output vectors close to the one hot vector for each letter. In the illustration, the element of the output vector that is shown in green should be maximized.

In practice, we delay the input with a buffer of 15 null-characters. This is necessary to allow the recurrent network to learn a valid starting state. We delay the output with a buffer of $15 + d$ null-character symbols. Here d represents the sequence delay which we tuned empirically, guided by the insight in Section III based on Figure 3. We pad each word with a suffix sequence of null characters if its length, after adding input delays, is below the maximum word length in the language.

V. Experiments

Our model contains 2 hidden layers of 512 units each. We train our model for 150 epochs. The percentage of erroneous words in the validation set corrected by the model increases and then hits a maximum at a certain epoch and starts to decrease. We use the model from the epoch that corrects the maximum number of erroneous words in the validation set and then employ the same model on test data. Interestingly, the model corresponding to this epoch also gives the maximum F-Score over the validation dataset across all epochs. For more details refer www.cse.iitb.ac.in/~rohitsaluja/IndicOCR.

For LSTM models, while increasing the delay between the input and output word sequence results in increased context, we found that when the delay is increased beyond a certain point, it also increases the error in the output of the network. Intuitively, this could be because a larger delay makes it

difficult for the model to predict the corrections and/or the model overfits on higher level n-grams. As stated earlier, we found a sequence delay of 5 to 7 (character) units between the input word and output word to work reliably in practice for the Indian Languages we work with.

We used the dataset, introduced in Section III. As stated earlier, we balanced our dataset for Hindi word pairs. This also ensures that we make a fair comparison with the state-of-the-art error detection results [14]. We use a train-val-test split ratio of 64-16-20 in our experiments.

A. Error Detection Experiment

For an input OCR word, if our trained LSTM model outputs a word different from the input, we mark the input word as incorrect and mark it as correct otherwise. This is how we detect errors using our model trained for error correction. The error detection results for our model are better than the results of the state-of-the-art system (see Section VI).

The measures used for error detection are defined below:

1) True Positives (TP) or Typing/Suggestion Efforts: Percentage of incorrect OCR words that are marked as incorrect. Such words would need typing corrections or suggestion selection (if suggestions are available).
2) True Negatives (TN) or Verification Gain: Percentage of correct OCR words that are marked as correct.
3) False Positives (FP) or Verification Efforts: 100-TN.
4) False Negatives (FN) or Unavoidable Accuracy Loss: 100-TP. Since the user tends to ignore such incorrect words, that are marked as correct, they lead to unavoidable accuracy loss.
5) Precision: TP/(TP+FP), Recall: TP/(TP+FN) and F-Score: Harmonic mean of Precision and Recall.

There is always a trade-off between TP and TN, though its degree might depend on the model. Use of conventional dictionary based methods increases the TP but lowers the TN. The reason is that a system that marks most incorrect words as incorrect also tends to mark several correct words as incorrect. Similarly, a conjoining rule-based method for increasing TN lowers TP. This trade-off can be captured through F-Score. Maximizing F-Score tends to yield models that are balanced in both these measures.

B. Error Correction Experiment

We evaluate the performance of our model for error correction against two baselines that we created based on combining the ideas of standard dictionary-based error correction with the back-off from n-gram character confusions of the OCR system. For each OCR word o, we find the set of closest words W from the dictionary V (*i.e.*, the vocabulary of known words). We compute the posterior distribution (1) on w to rank the replacement words in W.

$$w^* = \arg\max_{w \in W} P(w|o) = \arg\max_{w \in W} P(o|w)P(w) \qquad (1)$$

to determine the desired word w^*. For $P(w)$ we used word frequencies from training and validation datasets, while $P(o|w)$ is estimated based on character confusion probabilities as $\prod_{(c_o,c_w) \in C_{ow}} P(c_o|c_w)$. Here C_{ow} is the set of n-gram character confusions[1], (c_o, c_w), required to convert o into w. For $P(c_w|c_o)$ we consider the frequency of confusions in the union of training and validation datasets. We used Laplace smoothening for the unseen confusions.

In our first baseline, we consider V to be the set of ground truth words from training and validation datasets. The test dataset is different from V (exactly as in the LSTM model). We call this the lower baseline. In the second baseline, we assume that all the ground truth words for the test dataset are also available in V. Hence, we call it our upper baseline: an idealized, best possible baseline for word level corrections.

C. Suggestion Generation Experiment

We observe that four different contexts are helpful in correcting the characters of an OCR word;

1) PC: Preceding characters from the OCR word itself,
2) SC: Succeeding characters from the OCR word [2],
3) PCPW: Preceding characters from the OCR word & its preceding word neighbors and
4) PCSW: Preceding characters from the OCR word & its succeeding word neighbors.

For Sanskrit, which exhibits the highest proportion of OOV words (see Section III), we train the LSTM network with these 4 contexts & obtain a model from each. For PC, we trained the forward character model explained in Section IV. In contrast to this, for SC, we train another model with the reversed order of characters in both the input & the output words. For PCPW, we train another LSTM model that takes characters from 6 words in the input, 5 of which precede the present word. In all, 6 correct words are considered in the output. On test data, 6 OCR words are provided as input, while we are concerned only with the corrections made to the last word. Similarly, for PCSW, we train a model that considers the characters from 6 words in the input, 5 of which succeed the present word. To achieve this, we reversed the order of words used to train the LSTM. For last 2 models, we used the delay of 20 characters.

VI. Results

A. Error Detection Results

Lang.	TP	TN	FP	FN	Prec.	Recall	F-Score
San.	92.63	94.54	5.45	7.36	94.84	92.64	93.72
Mal.*	87.56	94.23	5.77	12.44	93.82	87.56	90.58
Mal.	92.62	96.02	3.98	7.38	93.26	92.63	92.94
Kan.	98.51	97.28	2.71	1.48	96.92	98.41	97.66
Hin.*	72.30	90.90	9.10	27.70	89.30	77.22	82.82
Hin.	91.96	93.86	6.14	8.04	92.94	91.95	92.44

TABLE II
Error Detection Results in Indic OCR. *State-of-the-art results [14]

[1]Computed using dynamic programming.

[2]Certain mistakes in Indian language scripts are more sensitive to succeeding characters than preceding ones.

We present our basic error detection results in Table II. Here, we note that the word-level error detection on OCR output obtained using the basic forward character level LSTM model outperforms the state-of-the-art [14] results (shown as Lang.*) in Malayalam and Hindi. Further, it is important to note that the results for Sanskrit and Kannada are better than the state-of-the-art results for Malayalam and Hindi respectively, although the former languages have the higher percentage of OOV words (see Figure 2).

B. Error Correction Results

Lan.	Word Error Rate (WER)				%age words corrected by		
	OCR	Baseline		LSTM	Baseline		LSTM
		Lower	Upper		Lower	Upper	
San.	51.20	58.60	20.01	21.41	9.62	66.12	63.34
Mal.	37.28	48.43	10.83	10.59	9.09	58.20	78.30
Kan.	47.44	48.13	27.77	15.73	18.31	54.57	69.66
Hin.	46.80	45.43	34.17	16.71	20.94	27.46	72.47

TABLE III
DECREASE IN WER AND PERCENTAGE OF ERRONEOUS WORDS AUTO CORRECTED BY OUR MODEL.

In Table III we compare our method against the two baseline models described in Section V. It can be observed that we achieve a decrease in overall WER by at least 26.7% & at least 63.3% of the erroneous words were corrected by our model for all the languages. Our LSTM-based model outperforms both the baseline models. Even though the upper baseline model contains the ground truth word for the test data in its dictionary, it is unable to correct all errors. This is because the word searched for in the vocabulary invariably results in a large set of neighbors, from which it is ambiguous to pick the correct word, even using the knowledge of OCR-specific n-gram confusions. Only in the case of Sanskrit, the idealized, upper baseline model is marginally better because the language has many long words and usually only one of words in the dictionary is a neighbor of the word searched for. It should be noted that the LSTM has no access to the ground truth of the test data, as is the case for the upper-bound baseline model. The poor performance of the lower baseline can be attributed to the fact that the vocabulary of ground truth words in the test data is significantly different from the training and validation sets.

Our LSTM-based model with a fixed delay also worked reliably on smaller datasets of $20k$ and $28k$ word pairs in Gujarati and Telugu respectively, which we obtained from the state-of-the-art [14]. We were able to correct 75.67% words in Gujarati & 73.36% words in Telugu, with F-Scores of 91.16 & 91.57 respectively. In both these languages, our models performed better than the upper baselines which corrected 58.65% & 49.22% words in Gujarati & Telugu respectively.

We test for statistical significance of the performance of our model over the upper baseline using a Wilcoxon Signed-Rank test. The null hypothesis is that our model does not perform better than the upper baseline. On the percentage of erroneous words corrected by both the methods, we obtain a significance of 3.8% for the 6 Indian languages mentioned above. This clearly rejects the null hypothesis and supports our claim.

In order to analyze the partial corrections in words, or character level improvements, in Figure 5 we present the histograms of edit distance of OCR words from ground truth words, and LSTM output from ground truth. Note, that these are based on test data. Our model reduces character level errors for most words. This is apparent from the shift in the histogram for OCR words with higher edit distances from the ground truth to lower values of edit distances.

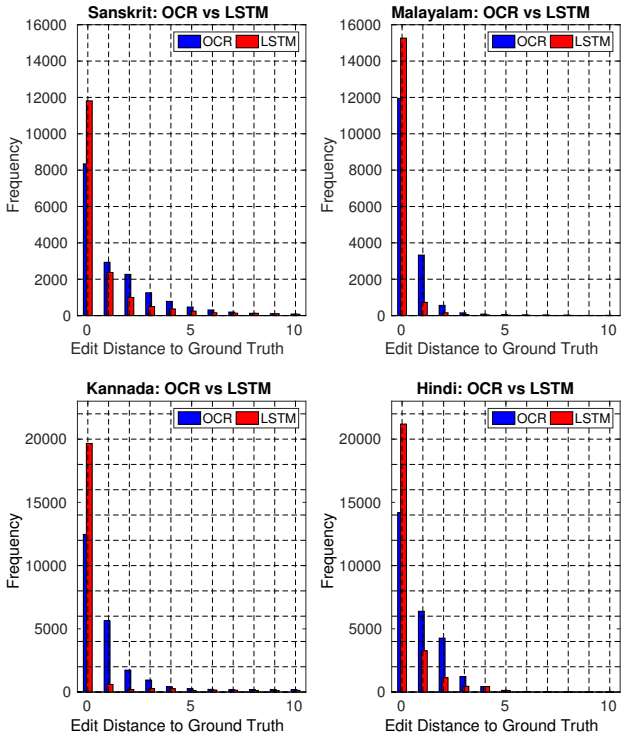

Fig. 5. Histogram of edit distance between OCR word and ground truth word pairs (in blue), LSTM output and ground truth word pairs (in red). As shown, many of the erroneous OCR words (with edit distance ≥ 1) get corrected to ground truth words (edit distance $= 0$) after passing through the LSTM. Also, there are many partial corrections made by the LSTM-based model and this is seen as a left shift in the red histograms as compared to the corresponding blue histograms.

Examples of words completely corrected by our system can be seen in Figure 1. In Figure 6, we show examples of words that are partially corrected by our model.

OCR Word	LSTM output	Correct Word
गौघ्रपरिधिभागा	शीघ्रपरिधिभागा	शीघ्रपरिधिभागाः
തേങ്ങാക്ക്യഷി	തേങ്ങാക്കൃഷി	മാങ്ങാക്കൃഷി
ದಂದದಲಿ	ದ್ವಂದ್ವದಲ್ಲಿ	ದ್ವಂದ್ವದಲ್ಲಿ
स्म?ल्याआं	समस्याओं	समस्याओं

Fig. 6. Examples of OCR words partially corrected by LSTM.

If OCR incorrectly map words from the document image to other correct words in the language, our model is unable to detect such Real Word Errors (RWE) and does not correct

these words. Examples are shown in Figure 7 (top).

OCR Word/LSTM output	Correct Word
शीघ्रकेन्द्रभुक्ति	शीघ्रकेन्द्रभुक्तिः
വേണമെന്നു	വേണമെന്നു്
ಗಾಯಾಳುವನು	ಗಾಯಾಳುವನ್ನು
उसने	उसके

OCR Word/Correct Word	LSTM output
समशङ्क	समशङ्कः
ശലയ്യം	സതയ്യം
ಮೊರೆಯಿಡಲಾರಂಭಿಸಿದವು	ಮೊರೆಯಿಡಲಾರಂಭಿಸಿದ್ದ
मुनीम	सुनीम

Fig. 7. Examples of OCR words not corrected (top) and corrupted (bottom) by the LSTM.

In a few rare cases, the LSTM does introduce new errors in words that are correct in the OCR output. This seems to happen when the model replaces less frequent n-grams in the word by more frequent n-grams. More training data containing the less frequent n-grams should be able to correct these errors. Some examples of these is shown in Figure 7 (bottom).

C. Suggestion Results

Suggestion Index	Context for training model	%age of correct suggestions	%age of unique suggestions
1.	PCPW	63.98	63.98
2.	PC	63.34	8.45
3.	SC	55.02	4.48
4.	PCSW	57.34	0.57

TABLE IV
PERCENTAGE OF ERRONEOUS WORDS CORRECTED BY MODELS TRAINED WITH DIFFERENT CONTEXTS IN SANSKRIT

As explained in Section V, we trained 4 additional models for Sanskrit, to generate suggestions for the correct word. In Table IV, we summarize the results, comparing the quality of the suggestion generated by each of these models. The last column in each row is for the percentage of correct suggestions achieved by the corresponding model, that could not be corrected by the models in the rows above them. We achieve an overall correct suggestion percentage for around 77% of erroneous words using the strategy of obtaining different suggestions for the same OCR word on the basis of different contexts. Using this user-in-the-loop system, we beat the upper baseline model for Sanskrit as well.

VII. CONCLUSIONS

In this paper, we demonstrate the use of LSTM with a delay, for jointly learning error patterns and language models. We show that these models are robust at detecting errors in OCR output and perform reliably in correcting them. We demonstrate the usefulness of our model by performing several experiments on the OCR texts of multiple Indian languages with varying scripts. We demonstrate performance better the state-of-the-art for error detection. As future work, it would be interesting to investigate the performance of BLSTM models and character level attention [7] models. We would also like to explore the possibility of training such models adaptively and on the fly to support the user while correcting the OCR output in Indic documents.

Acknowledgements: We thank NVIDIA for their GPU hardware support.

REFERENCES

[1] M. Cheriet, N. Kharma, C.-L. Liu, and C. Suen, *Character Recognition Systems: A Guide for Students and Practitioners*. John Wiley & Sons, 2007.

[2] I. Kissos and N. Dershowitz, "OCR Error Correction Using Character Correction and Feature-based Word Classification," in *12th IAPR Workshop on Document Analysis Systems (DAS)*, 2016, pp. 198–203.

[3] J. Evershed and K. Fitch, "Correcting Noisy OCR: Context Beats Confusion," in *Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage*, 2014, pp. 45–51.

[4] ICDAR, "Competition on Post-OCR Text Correction," https://sites.google.com/view/icdar2017-postcorrectionocr/. Last accessed on April 8, 2017.

[5] N. Sankaran and C. Jawahar, "Error Detection in Highly Inflectional Languages," in *Document Analysis and Recognition (ICDAR), 12th International Conference on*, 2013, pp. 1135–1139.

[6] I. Sutskever, J. Martens, and G. E. Hinton, "Generating Text with Recurrent Neural Networks," in *Proceedings of the 28th International Conference on Machine Learning (ICML)*, 2011, pp. 1017–1024.

[7] Z. Xie, A. Avati, N. Arivazhagan, D. Jurafsky, and A. Y. Ng, "Neural Language Correction with Character-based Attention," *arXiv preprint arXiv:1603.09727*, 2016.

[8] K. Kukich, "Techniques for Automatically Correcting Words in Text," *ACM Computing Surveys (CSUR)*, vol. 24, no. 4, pp. 377–439, 1992.

[9] Y. Bassil and M. Alwani, "OCR Context-sensitive Error Correction Based on Google Web 1T 5-gram Data Set," *arXiv preprint arXiv:1204.0188*, 2012.

[10] A. Carlson and I. Fette, "Memory-based Context-sensitive Spelling Correction at Web Scale," in *International Conference on Machine learning and applications (ICMLA)*, 2007, pp. 166–171.

[11] A. Wilcox-O'Hearn, G. Hirst, and A. Budanitsky, "Real-Word Spelling Correction with Trigrams: A Reconsideration of the Mays, Damerau, and Mercer Model," in *International Conference on Intelligent Text Processing and Computational Linguistics*, 2008, pp. 605–616.

[12] A. R. Golding and Y. Schabes, "Combining Trigram-based and Feature-based Methods for Context-sensitive Spelling Correction," in *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, 1996, pp. 71–78.

[13] R. Smith, "Limits on the Application of Frequency-based Language Models to OCR," in *International Conference on Document Analysis and Recognition (ICDAR)*, 2011, pp. 538–542.

[14] V. Vinitha and C. Jawahar, "Error Detection in Indic OCRs," in *12th IAPR Workshop on Document Analysis Systems (DAS)*, 2016, pp. 180–185.

[15] K. Nair and C. Jawahar, "A Post-Processing Scheme for Malayalam Using Statistical Subcharacter Language Models," *Proceeding of the IAPR Workshop on Document Analysis Systems (DAS)*, pp. 363–370, 2010.

[16] G. Lehal, C. Singh, and R. Lehal, "A Shape Based Post Processor for Gurmukhi OCR," in *International Conference on Document Analysis and Recognition (ICDAR)*, 2001, pp. 1105–1109.

[17] U. Pal, P. K. Kundu, and B. B. Chaudhuri, "OCR Error Correction of an Inflectional Indian Language Using Morphological Parsing," *Journal of Information Science and Engg.*, vol. 16, no. 6, pp. 903–922, 2000.

[18] M. Choudhury, M. Thomas, A. Mukherjee, A. Basu, and N. Ganguly, "How Difficult is it to Develop a Perfect Spell-checker? A Cross-linguistic Analysis through Complex Network Approach," *arXiv preprint physics/0703198*, 2007.

[19] Google, "Google's Optical Character Recognition (ocr) Software Works for 248+ Languages," https://opensource.com/life/15/9/open-source-extract-text-images. Last accessed on March 10, 2017.

[20] I. Z. Yalniz and R. Manmatha, "A Fast Alignment Scheme for Automatic OCR Evaluation of Books," in *International Conference on Document Analysis and Recognition (ICDAR)*, 2011, pp. 754–758.