Sub-word Embeddings for OCR Corrections in Highly Fusional Indic Languages

Rohit Saluja*
IITB-Monash Research Academy
Mumbai, India

Mayur Punjabi*
IIT Bombay
Mumbai, India

Mark Carman†
Politecnico di Milano
Milan, Italy

Ganesh Ramakrishnan*
IIT Bombay
Mumbai, India

Parag Chaudhuri*
IIT Bombay
Mumbai, India

*{rohitsaluja, mayurpunjabi, ganesh, paragc}@cse.iitb.ac.in, † mark.carman@polimi.it

Abstract*—Texts in Indic Languages contain a large proportion of out-of-vocabulary (OOV) words due to frequent fusion using conjoining rules (of which there are around 4000 in Sanskrit). OCR errors further accentuate this complexity for the error correction systems. Variations of sub-word units such as n-grams, possibly encapsulating the context, can be extracted from the OCR text as well as the language text individually. Some of the sub-word units that are derived from the texts in such languages highly correlate to the word conjoining rules. Signals such as frequency values (on a corpus) associated with such sub-word units have been used previously with log-linear classifiers for detecting errors in Indic OCR texts. We explore two different encodings to capture such signals and augment the input to Long Short Term Memory (LSTM) based OCR correction models, that have proven useful in the past for jointly learning the language as well as OCR-specific confusions. The first type of encoding makes direct use of sub-word unit frequency values, derived from the training data. The formulation results in faster convergence and better accuracy values of the error correction model on four different languages with varying complexities. The second type of encoding makes use of trainable sub-word embeddings. We introduce a new procedure for training fastText embeddings on the sub-word units and further observe a large gain in F-Scores, as well as word-level accuracy values.

I. Introduction

The procedure for extracting the text present in images of documents is referred to as Optical Character Recognition (OCR). Due to variations in fonts, occlusions, inflections, and/or the scarcity of training data, every OCR system produces some errors in the text it recognizes. It is therefore important to detect errors and provide corrections or (in the case of historical book scanning) suggestions to help the human annotators. There have been various works published on OCR error correction [1], [2] and the importance of post-OCR error correction is further underlined by competitions on historical documents in English, French, German, Finish, Spanish, Dutch, Czech, Bulgarian, Slovak and Polish [3], [4].

For inflectionally rich languages, standard methods for spell-checking using a dictionary look-up produce inadequate results for the task of OCR correction [5]. Due to the use of word conjoining rules in such languages, off-the-shelf dictionaries are substantially incomplete [6], [7]. Recent work on error correction in medical documents tries to solve the large (but nonetheless limited) vocabulary problem via nearest neighbor search in an embedding space [8]. Such techniques are not as effective on Indic languages due to the problems

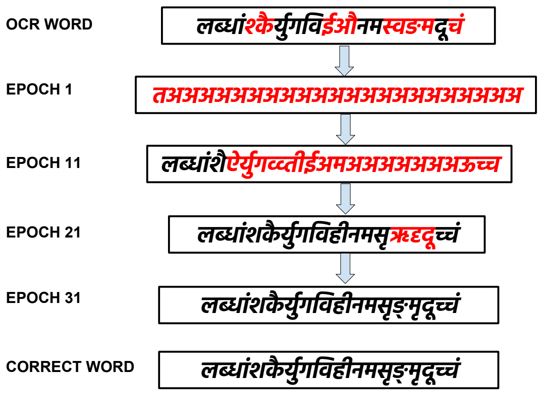

OCR WORD
लब्धांश्कैर्युगविईऔनमस्वङ्मदूचं
EPOCH 1
तअअअअअअअअअअअअअअअअअअअअअअ
EPOCH 11
लब्धांशैऐर्युगव्व्तीईअमअअअअअअऊच्च
EPOCH 21
लब्धांशकैर्युगविहीनमसृऋद्दूच्चं
EPOCH 31
लब्धांशकैर्युगविहीनमसृङ्मृदूच्चं
CORRECT WORD
लब्धांशकैर्युगविहीनमसृङ्मृदूच्चं

Fig. 1. Correction flow of our model for a complex word in Sanskrit.

of lack of language-specific text resources, incomplete vocabularies and word ambiguities that arise due to a high degree of branching in spell-net [5]–[7], [9]. The problem of word ambiguities can be further appreciated by observing the poor performance of skylines with respect to the model described in Saluja et al. [7]. Such skylines involve nearest neighbor search in the vocabulary of correct words available in both training and test data. Although, such skylines also invoke OCR-specific confusions to break ties, such skylines are able to auto-correct only around 66% and 27% of OCR errors in Sanskrit (Indic language with high number of inflections) and Hindi [9] (Indic language with fewer inflections but higher word ambiguities) respectively.

For Indic Languages, the conjoining rules are of two types, *viz.*, simple and complex. The simple rules involve the agglutination of valid word forms, just as the words "can" and "not" form the word "cannot" in English. The complex rules involve the fusion of joined words, analogous to the formation of "coudn't" by joining the words "could" and "not" in English. In Indic languages, often more than two words are conjoined using such simple agglutinative and complex fusional rules. An example of such a conjoined word is illustrated in Figure 1. Even if "wouldn't" was an out-of-vocabulary (OOV) word, the errors in OCR text corresponding to such a word could be corrected based on the sub-word units derived from another vocabulary word ("couldn't"). This language phenomenon forms the basis of our work. As a baseline, we use the frequency of a sub-word unit (of a conjoined OCR word) in a language corpus to correct the characters in the OCR output.

Some word embeddings also include sub-word units, such as in fastText [10], ELMO [11] and BERT [12]. Of these we find fastText to be most naturally suited for the purpose of pre-training based on reconstruction of each sub-word using its context in fixed length sub-strings (refer Figures 2, 3). Such an embedding helps in representing the information related to the frequency and context of sub-words in the language and especially benefits OOV words at test time. This makes fastText a good fit for the segmentation of the complex conjoined words into generally used words for Sanskrit [13].

Sub-word units, such as n-grams and their (possibly noisy) contexts, can be extracted from the OCR output text, and compared with their frequency statistics over a general text corpus. Different statistical functions for the sub-word units can be computed, with text embeddings, such as fastText, essentially learning one such combined function of n-grams & text dependency from the language data. For the OCR correction task, we will consider the frequencies of sub-words in the ground truth of training data as a baseline approach and compare it with modified fastText embeddings (explained in Section III) as an alternative approach.

II. Related Work

There have been several attempts at using dictionaries to correct errors in text [14]–[16]. The methods are not reliable for languages with agglutinations and fusions as their performance suffers due to the OOV problem. Approaches that perform corrections based on context n-grams are more effective. The work by Wilcox-OHearn et al. [17] uses a trigram-based noisy-channel model. Bayesian methods based on part-of-speech trigrams have also been explored for corrections based on context [18]. The work by Smith [19] further concludes that noisy-channel models that closely model the underlying classifier and segmentation errors are required by OCR systems to improve performance.

Recent work on neural language correction [20] has shown the benefits of using Recurrent Neural Networks (RNN) for the purpose of correcting text. A Long Short Term Memory (LSTM) model has also been used in a recent work for post OCR corrections [7]. LSTM models can remember longer term context of the input sequence and are therefore quite successful in correcting OCR induced errors. This model outperforms all other models for post-OCR correction tasks in four Indic Languages with varying complexities. Error detection for Indic languages presents singular chal- lenges such as large unique word lists, lack of linguistic resources and lack of reliable language models [5]. There are many examples of work in the literature that focus on post-OCR corrections for a specific Indic language. The various techniques used include morphological parsing for Bangla [21], shape-based statistics for Gurmukhi [22] and a multi-stage graph module with a sub-character model for Malayalam [23].

Error detection in Indic languages using a conventional lookup technique is outperformed by the use of a Support Vector Machine (SVM) classifier [5]. This is further improved upon by using Gaussian Mixture Models (GMMs) and RNNs to detect errors in four Indic languages [6]. Recent work also presents a copying mechanism for post-OCR corrections in romanised Sanskrit [24].

Language	# Word Correction Pairs	μ, σ of Word Length	OOV percentage in test set	Word Error Rate for OCR (OOVs)
Sanskrit	86 k	10.22, 7.98	44.77	51.20 (72.16)
Malayalam	107 k	9.32, 4.93	49.32	38.32 (45.86)
Kannada	118 k	8.42, 3.86	26.59	47.44 (60.14)
Hindi	134 k	5.29, 2.53	23.70	46.80 (48.47)

TABLE I
Datasets used for our experiments.

The complexities involved in creating effective spell-checkers for Hindi, Bangla and English are discussed by Choudhury et al. [9] in terms of spell-nets. A spell-net is a graph with words as nodes and edit-distance based weighted edges. The spell-net for Hindi and Bangla have a higher average degree as compared to English. This causes a higher number of inter-word ambiguities in Indic Languages. Moreover, in the spell-nets for Indic Languages, the correlation between degrees of neighboring words (nodes) is higher than English. This makes it more difficult to rank the candidate suggestions for an incorrect word whenever the word has a high degree in the spell-net. This work improves the performance of the current best model in literature for post-OCR corrections in Indic text [7] by the use of sub-word embeddings. We present our datasets and methods to derive these sub-word embeddings in Section III. Our model is described in Section III-C. The details of our experiments are provided in Section IV. We present our results and conclusions in Sections V and VI respectively. In summary, our key contributions are as follows:-

1) We show that sub-word level information, derived from language or training data itself, can help to correct OCR errors in Indic Languages.

2) We demonstrate that augmenting the input of LSTM models with the frequencies of OCR sub-words in the language data, performs as well as using fastText embed- dings (trained on same data) for correcting Indic OCR.

3) We propose a new procedure for training fastText on sub- word units present in the constant length substrings. This involves the transformation of language data in such a way that it not only includes all the substrings within the language, but also retains the character level context of substrings in the language. This method is shown to improve the accuracy as well as computational performance (see Fig. 1) for error correction in Indic OCR with respect to the state-of-the-art and the baseline models.

III. Datasets and Methodology

We work on four Indian languages with varying complexities. Table I summarizes the languages and the details of the dataset that we obtain from the state-of-the-art work [7]. For Malayalam, we obtain 81k pairs of OCRed words and their corresponding corrected versions (hereafter referred to as *correction pairs*). We also include the 26k correction pairs obtained from a previous work [6]. Putting these together,

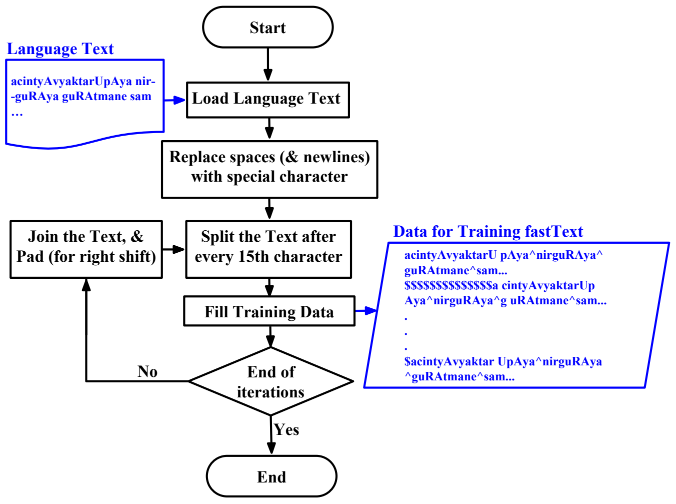

Start
Language Text
acintyAvyaktarUpAya nir-
-guRAya guRAtmane sam
...
Load Language Text
Replace spaces (& newlines) with special character
Join the Text, & Pad (for right shift)
Split the Text after every 15th character
Data for Training fastText
acintyAvyaktarU pAya^nirguRAya^
guRAtmane^sam...
$$$$$$$$$$$$$$a cintyAvyaktarUp
Aya^nirguRAya^g uRAtmane^sam...
.
.
.
$acintyAvyaktar UpAya^nirguRAya
^guRAtmane^sam...
Fill Training Data
No
End of iterations
Yes
End

Fig. 2. Flowchart for transformation of language data (shown in SLP1 for illustration) for training fastText on all possible substrings of fixed-length (15 here). The transformation retains the context of each substring in the language.

we get a total of 107k correction pairs for Malayalam. As shown in Table I, the dataset consists of the order of 100,000 pairs of “OCR word, Ground Truth word” in each of the four languages. The mean and standard deviations of word lengths in Table I for the different languages are consistent with the fact that Sanskrit has the highest number of word conjoining rules, followed by Malayalam, Kannada and Hindi. The average word length as well as the standard deviation of word lengths is also highest for Sanskrit and both measures decrease with the reduction in complexity of languages as shown in Table I. Further note that although the OOVs in the Malayalam test set (w.r.t. to training and validation set) are higher in percentage as compared to Sanskrit, still the Word Error Rates (WER) for general OCR words as well as OOV words in Sanskrit are higher than that of Malayalam.

We work on the task of correcting the (possibly incorrect) character x_t in the OCR word $x_{1:T_x}$ to the correct character y_t. Owing to confusions involving multiple characters in the source and/or target, some of the x_t and y_t could be blanks. The sub-word context $x_{t-l:t+l}$, (with l being a hyper-parameter) can be further utilized to predict the correct character y_t. The input space of our model (explained in Section III-C) can explode if we provide the sub-word units directly in the form of one-hot-encodings (OHE). We therefore provide the information about the context sub-words in the form of normalized frequencies for the baseline, and fastText embedding vectors trained with a new procedure that we describe later in this section.

A. *Baseline: Sub-word units based learning*

As a baseline, for each input character x_t in the training sample $\{x_{1:T_x}, y_{1:T_y}\}$, we extract the bag of sub-words with lengths varying from 2 to $l+1$ from a context window of length $2l+1$ around x_t. We only consider the sub-word units that contain the character x_t. Thus the sub-word units that we consider for the t^{th} character in the OCR word $x_{1:T_x}$ are: 2-grams ($\{x_{t-1:t}\}$, $\{x_{t:t+1}\}$), 3-grams ($\{x_{t-2:t}\}$, $\{x_{t-1:t+1}\}$, $\{x_{t:t+2}\}$), *etc.*, up to (l+1)-grams ($\{x_{t-l:t}\}$, $\{x_{t-l+1:t+1}\}$, ... , $\{x_{t:t+l}\}$).

We find the normalized frequency of each sub-word unit in the ground truth of the training data Y and augment sub-word frequencies to our model as described in Section III-C.

B. *A new procedure for training fastText on sub-word units*

As an important contribution in this paper, we provide a new procedure for training embeddings such as fastText for the task of Indic OCR corrections [10]. The training procedure is driven by and based on the observations described in the previous sub-section. While training the fastText embeddings, we consider the substrings of length $2l+1$ in the language. We split the language text at every $2l+1^{th}$ character (including space characters) to form substrings of length $2l+1$, before learning the desired embedding over the entire string. It is important to emphasize that the fastText implementation involves the bag of smaller sub-words (of length 2 to $l+1$ for our case) within the substring (of length $2l+1$) that we obtain. This is similar to the learning described in the previous sub-section. The flowchart for this process along with an example of language data in SLP1 (Sanskrit Library Phonetic Basic encoding scheme) format and corresponding training data obtained using fixed length substrings of size $2l+1 = 15$ are illustrated in Figure 2. As shown, we first replace each space (and newline) character in the language text with a special character. We then split the data every 15 characters to form substrings of length 15 (adequately padding the end of language text). We then iteratively 1) pad the language text in such a way that the substrings starting from the subsequent characters are considered, and 2) repeat the above splitting, 14 more times to include every possible sub-word of length 15 in the language. This transformation also retains adequate context for each substring in the original language text.

C. *Model*

We train our models for OCR corrections using the global information about our training data, in the form sub-word frequency values as well as fastText embedding vectors derived from the ground truth of training data as shown in Figure 3.

Since the OCR output text is mostly correct (i.e. more characters are correct than incorrect), and since the errors in such texts follow certain confusion patterns (which arise due to similarly shaped glyphs in the language font), for OCR correction character-based approaches are generally used [4], [7]. To predict the t^{th} character y_t of the correct sequence $y_{1:T_y}$ of length T_y, based on the OCR character sequence $x_{1:t+d}$, an LSTM with fixed sequential delay can be used [7]. Here d is context ahead of the location t in the input sequence $x_{1:T_x}$ required to resolve the error at that location. For the baseline, a database D is filled with the mapping of all possible sub-words with their frequency in the ground truth of training data as shown in Figure 3 (orange in bottom right). Then, we append the one hot encoding (OHE) of each character x_t at the input with the frequencies (which are derived from the mappings in the database D) of sub-words $x_{t-a:t+b}$ s.t.

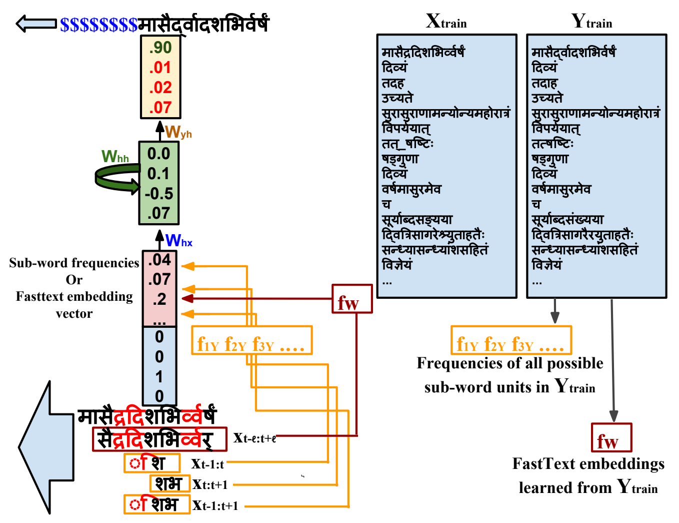


Fig. 3. An LSTM model is shown with 7 units of hard-wired delay (represented by the blue $), having 1 hidden layer of 4 units. The baseline flow is shown in orange and the sub-word embedding flow in dark red. For the baseline, the frequencies of all possible sub-words in the ground truth labels of training data are stored (f_{1Y},f_{2Y},f_{3Y} . . .) and the frequencies of the sub-words are derived from the stored frequencies. For the sub-word embeddings, the fastText embeddings are learned from the sub-words in training data (f_W). The fastText vector for the (sub-words present in) substring in context window is derived from these learned embeddings.

$0 \leq a, b \leq l$ within a context $x_{t-l:t+l}$ around it (orange in bottom left). For illustration, only three sub-word units within the context $x_{t-l:t+l}$ with l = 5 is shown in Figure 3. In comparison, for using fastText sub-word embeddings in our model, we first train fastText on (the sub-words present in) the constant length $(2l + 1)$ substrings as explained in Section III (shown in dark red in Figure 3). We then concatenate the one hot encoding of each character at the input with the embedding vector of the substring (of length $2l+1$) present in the context window $t - l : t + l$.

IV. Experiments

The datasets that we use are explained in Section III. We split our datasets as per the ratio 64:16:20 for training, validation and testing respectively. Following the literature, we evaluate on two tasks, *viz.*, error detection and error correction [7]. Inspired by the state-of-the-art work, we use the sequence delay of 7 for Sanskrit and Kannada, and 5 for Malayalam and Hindi [7]. We use the context of $l = 7$ on each side of the OCR character to derive the sub-word units, based on the n-gram level features used for training log-linear classifiers in the literature [25]. Thus, we effectively use the sub-words of length 2 to 8 for our baseline model. For the fastText based models, we use the embedding size of 100 while training the fastText with (sub-word units present in) constant length substrings $x_{t-l:t+l}$ (refer Section III-C). We train the fastText models for 100 epochs on the ground truth of each language dataset. As discussed earlier, we switched off the word level n-grams and use the sub-words of length 2 to 8 within context $x_{t-l:t+l}$. We use 2×512 sized hidden layer LSTM for all our experiments. The gradient descent algorithm, with the learning rate of 0.002 and decay of 0.97 after 10 epochs of training, is used. We train our models with the cross-entropy loss for 200 epochs.

We use the disagreements between the input and output of our model for error detection. It is important to note that such methodology also allows us to colour code the errors at the granularity of characters as shown in Figure 3. We use F-Scores as the measure for error detection. Our models naturally learn to correct errors as we train them on the pairs of OCR word and its ground truth. We use Word Error Rates (WER) of the output of our models w.r.t. ground truth data, and percentage of erroneous words corrected by our model as the measures for error correction.

Method	TP	TN	FP	FN	Prec.	Rec.	F-Scr.
Normal training (Wikipedia & web)	93.74	94.19	5.80	6.25	94.57	93.74	94.16
Normal training (our data)	94.90	94.61	5.39	5.10	95.01	94.90	94.95
New training procedure (our data)	**95.11**	**95.39**	**4.61**	**4.89**	**95.71**	**95.11**	**95.41**

TABLE II
Effect of pre-training fastText with different datasets and effect of training procedure in Sanskrit.

V. Results

A. Error Detection Results

Here, we first discuss the effect of training our model with different fastText embeddings, then we discuss the results on two different encodings described in Section III across various languages.

1) Effect of training LSTM with different fastText embeddings: It is important to note that the average word length and standard deviation in word length is highest for the Sanskrit dataset as shown in Table II. Thus, for Sanskrit, we perform our experiments with fastText embeddings trained on 1) a large generalized corpus, 2) our data, and 3) our data with new training procedure explained in Section III. The results are shown in Table II. As shown in rows 1 and 2, the FScore of model with fast-text embeddings pre-trained on the ground truth of our our data is better than the model with fastText embeddings pre-trained on large amount of general data. This happens probably because there is sharing of sub-words (or domain) or OCR confusions among the training and the test datasets. However, when we train fastText on the ground truth of our training data with new training procedure, and use the pre-trained embeddings derived from it with our model, the results outperform the other methods as shown in the 3^{rd} row of Table II. This strongly supports our claim of contributing a novel, useful training methodology using sub-word embeddings.

2) Results on different Languages: We perform experiments for all the languages with the baseline model, and the fastText embeddings, both trained on the ground truth of our training data. It is important to note that the state-of-the-art

Lang.	TP	TN	FP	FN	Precision	Recall	F-Score
Sanskrit (OHE only)*	92.63	94.54	5.45	7.36	94.84	92.64	93.72
Sanskrit (Baseline: OHE and sub-word frequencies)	94.49	95.20	4.79	5.51	95.52	94.49	95.02
Sanskrit (Final: OHE and fastText with new training procedure)	**95.11**	**95.39**	**4.61**	**4.89**	**95.71**	**95.11**	**95.41**
Malayalam (OHE only)	91.40	96.39	3.61	8.60	94.02	91.40	92.69
Malayalam (Baseline: OHE and sub-word frequencies)	91.62	**96.42**	**3.58**	8.37	**94.08**	91.62	92.84
Malayalam (Final: OHE and fastText with new training procedure)	**94.70**	95.77	4.23	**5.30**	93.29	**94.70**	**93.99**
Kannada (OHE only)*	98.40	97.18	2.82	1.60	96.92	98.41	97.66
Kannada (Baseline: OHE and sub-word frequencies)	**98.64**	96.66	3.34	**1.36**	96.38	**98.64**	97.50
Kannada (Final: OHE and fastText with new training procedure)	98.36	**97.53**	**2.47**	1.64	**97.29**	98.36	**97.82**
Hindi*	91.96	93.86	6.14	8.04	92.94	91.95	92.44
Hindi (Baseline: sub-word frequencies)	93.68	94.36	5.64	6.32	93.60	93.68	93.64
Hindi (Final: OHE and fastText with new training procedure)	**96.92**	**95.68**	**4.32**	**3.07**	**95.18**	**96.92**	**96.04**

TABLE III
ERROR DETECTION RESULTS IN INDIC OCR. *STATE-OF-THE-ART RESULTS [7]. IT IS IMPORTANT TO NOTE THAT DATA-SETS USED FOR DERIVING SUB-WORD FREQUENCIES AND FASTTEXT EMBEDDINGS ARE SAME IN ALL THE SETTINGS FOR EACH LANGUAGE.

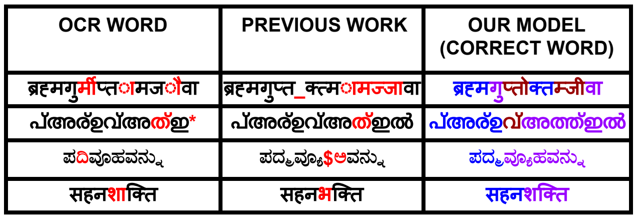
Lan.	Word Error Rate (WER)		%age words
	OCR (OOVs)	LSTM (OOVs)	corrected by LSTM
San.*		21.41 (32.67)	63.34
San.(Baseline)	51.20 (72.16)	17.85 (**28.03**)	70.05
San.(Final)		**17.72** (28.71)	**70.13**
Mal.		11.83 (**16.07**)	75.22
Mal.(Baseline)	38.32 (45.86)	11.55 (16.30)	75.60
Mal.(Final)		**10.95** (17.28)	**78.24**
Kan.*		15.73 (25.71)	69.66
Kan.(Baseline)	47.44 (60.14)	15.53 (27.08)	70.30
Kan.(Final)		**15.38** (**25.32**)	**70.90**
Hin.*		16.71 (29.23)	72.47
Hin.(Baseline)	46.80 (48.47)	14.42 (26.29)	75.59
Hin.(Final)		**9.58** (**20.26**)	**84.42**

TABLE IV
DECREASE IN WER AND PERCENTAGE OF ERRONEOUS WORDS CORRECTED BY OUR MODEL. *STATE-OF-THE-ART RESULTS [7].

results were already above 92% in terms of F-Score. Given that it can be challenging to further improving the F-Scores, we analyze the percentage improvement in F-Scores w.r.t. the state-of-the-art here. As shown in Table III our baseline model as well as the model based on fastText outperform the state-of-the-art results for all our experiments (except for Kannada where the results of our baseline are slightly lower than previous work). The improvements are because we provide the context information with each OCR character in the form of sub-word frequencies or sub-word embeddings as explained in Section III. As shown in the 2^{rd} row of Table III, our baseline model, that works on the principle of frequencies derived from sub-words in the ground truth of training data, performs as good as the fastText embeddings trained on the same data (refer row 2 of Table II) for Sanskrit. Our experiments show that there are gains of 1.38% and 1.80% F-Score using our baseline model and fastText based model respectively over the state-of-the-art for Sanskrit. Furthermore, the percentage increase in F-Scores for Malayalam, Kannada and Hindi are 0.43%, -0.16% and 1.30% respectively when we use our baseline model, and 1.62%, 0.16% and 3.69% when using fastText embeddings pre-trained with the proposed procedure. We further noted that all our models converge to 90% of F-Score (on test set) within the first 20 epochs of training. Thus we gain both higher performance and faster convergence with the pre-trained (or pre-calculated) encodings.

B. Error Correction Results

In Table IV we show that for Sanskrit, our baseline model reduces the word level errors to 17.85%, which is 3.56% better



OCR WORD	PREVIOUS WORK	OUR MODEL (CORRECT WORD)
ब्रह्मगुर्मीप्त◌ामज◌ौवा	ब्रह्मगुप्त_क्तम◌ामज्जावा	ब्रह्मगुप्तोक्तम्जीवा
പ്അര്ഉവ്അത്ഇ*	പ്അര്ഉവ്അത്ഇൽ	പ്അര്ഉവ്അത്ത്ഇൽ
ಪದಿವೂಹವನ್ನು	ಪದ್ಮವ್ಯೂ$ಅವನ್ನು	ಪದ್ಮವ್ಯೂಹವನ್ನು
सहनशाक्ति	सहनभक्ति	सहनशक्ति

Fig. 4. The first column shows the OCR word with errors marked in red. The second column shows the output from previous work [7]. The final column shows corrected output from our system. These examples are words with agglutination (marked in blue-purple) & fusion (in dark red) in 4 Languages.

as compared to state-of-the-art results. The increase in the percentage of erroneous words corrected by our baseline model is 6.71% w.r.t. the state-of-the-art. The gains increase further with our final model based on pre-trained fastText embedding. The corresponding improvements with our baseline models in Malayalam, Kannada and Hindi are 0.28%, 0.30%, & 2.29% in terms of reduction in WER, and are 0.28%, 0.64%, & 2.82% in terms of percentage of erroneous words corrected by the models. For the final model (fastText trained with the proposed procedure), the corresponding reductions in WER are 0.88%, 0.35% and 7.13% respectively, and the gains in word correction are 3.02%, 1.24% and 11.95% respectively. Moreover, it can be observed that all the models consistently reduce the errors in OOV words. As shown in Table I, for Sanskrit, Hindi and Kannada, the higher context in form of sub-word information helps in reducing word error rates in OOVs w.r.t previous work [7]. Interestingly, for Malayalam, we observed slightly higher word error rates for OOVs w.r.t. baseline model. We conjecture that this is due to differences in the statistics for OOV words between the training and test set, due to the addition of data from a different source [6] (explained in Section III) to the test set. We also noted that the correction pairs from this dataset form the 35% of our test set which leads to a high percentage of OOVs in Malayalam w.r.t. Sanskrit as shown in Table I.

Figure 4 shows sample errors corrected by our model in different languages. The OCR words are shown in the first column. The corrections performed by the previous state-of-the-art model are shown in the second column [7]. The correct predictions of our model are shown in the third column. Here, the correct word forms in the language are shown in blue

OCR CONFUSIONS	FREQUENCY OCR OUTPUT	FREQUENCY PREVIOUS WORK	FREQUENCY OUR MODEL
ो -> ी	756	24	9
क्ष -> च	382	14	9
स्व -> ख	319	5	4
ല് -> ൽ	1087	52	28
ര് -> ർ	966	57	22
ന് -> ൻ	714	36	19
ಶ್ -> ರ್	184	12	5
ಖ -> ಬ	155	10	5
ಯ -> ಇ	126	4	1
े -> ं	707	1	0
थ -> य	600	19	17
ै -> ं	373	3	2

Fig. 5. Top 3 OCR confusions (Correct → OCR) in Sanskrit, Malayalam, Kannada & Hindi (Top-to-bottom).

and purple colors. Thus there is a change of color from blue to purple (or reverse) when two words are agglutinated. The fusions are shown in dark red color. As shown, our model is able to correct the highly complex words that involve agglutinations and/or fusions in different Indian Languages.

C. Analysis

We now substantiate how our model improves the detection/correction for top character confusions (and also improves over the previous model [7]) in the OCR output as motivated earlier in Section III-C. As shown in Figure 5, the previous model is able to reduce the confusions to a large extent in our test data, and our model reduces them further. It is important to note that these confusions are corrected by the models based on their context in different OCR sequences.

VI. Conclusion

We work on the task of error correction in four different Indian languages with varying complexities. As a baseline, we use sub-words within a context window around the OCR characters to be corrected. We append the OHE input of the LSTM model with the frequency of such sub-words in the ground truth of training data. Such a baseline outperform the state-of-the-art models for three Indic Languages. We further experiment with the concatenation of fastText embedding vectors, pre-trained on different datasets, with the OHE input of LSTM. We present that our baseline model works similar to the fastText embeddings when pre-trained on the same data from which we derive the frequencies for the baseline model. We also present a better procedure of training fastText with all possible substrings of the desired length. Such model outperforms our baseline models, in addition to state-of-the-art, for the four Indic Languages with varying complexities.

References

[1] I. Kissos and N. Dershowitz, “OCR Error Correction Using Character Correction and Feature-based Word Classification,” in *Proceedings of DAS*, 2016, pp. 198–203.

[2] 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.

[3] G. Chiron, A. Doucet, M. Coustaty, and J.-P. Moreux, “Icdar2017 competition on post-ocr text correction,” in *ICDAR*, vol. 1. IEEE, 2017, pp. 1423–1428.

[4] ICDAR, “Competition on Post-OCR Text Correction,” https://sites.google.com/view/icdar2019-postcorrectionocr. Last accessed on June 30, 2019.

[5] N. Sankaran and C. Jawahar, “Error Detection in Highly Inflectional Languages,” in *Proceedings of ICDAR*, 2013, pp. 1135–1139.

[6] V. Vinitha and C. Jawahar, “Error Detection in Indic OCRs,” in *Proceedings of DAS*, 2016.

[7] R. Saluja, D. Adiga, P. Chaudhuri, G. Ramakrishnan, and M. Carman, “Error detection and corrections in Indic OCR using LSTMs,” *Proceedings of ICDAR)*, 2017.

[8] P. Fivez, S. Šuster, and W. Daelemans, “Unsupervised context-sensitive spelling correction of clinical free-text with word and character n-gram embedding,” in *16th Workshop on Biomedical Natural Language Processing of the Association for Computational Linguistics*, 2017.

[9] 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 physics/0703198*, 2007.

[10] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, “Enriching word vectors with subword information,” *Transactions of ACL*, vol. 5, pp. 135–146, 2017.

[11] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, “Deep contextualized word representations,” in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2018, pp. 2227–2237.

[12] J. Devlin, M. Chang, K. Lee, and K. Toutanova, “BERT: pre-training of deep bidirectional transformers for language understanding,” *CoRR*, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[13] V. Reddy, A. Krishna, V. D. Sharma, P. Gupta, P. Goyal *et al.*, “Building a word segmenter for sanskrit overnight,” *arXiv:1802.06185*, 2018.

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

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

[16] 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.

[17] 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.

[18] 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 of ACL*, 1996, pp. 71–78.

[19] R. Smith, “Limits on the Application of Frequency-based Language Models to OCR,” in *Proceedings of ICDAR*, 2011, pp. 538–542.

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

[21] 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.

[22] G. Lehal, C. Singh, and R. Lehal, “A Shape Based Post Processor for Gurmukhi OCR,” in *Proceedings of ICDAR*, 2001, pp. 1105–1109.

[23] K. Nair and C. Jawahar, “A Post-Processing Scheme for Malayalam Using Statistical Subcharacter Language Models,” *Proceeding of DAS*, 2010.

[24] A. Krishna, B. P. Majumder, R. S. Bhat, and P. Goyal, “Upcycle your ocr: Reusing ocrs for post-ocr text correction in romanised sanskrit,” *arXiv:1809.02147*, 2018.

[25] D. Adiga, R. Saluja, V. Agrawal, G. Ramakrishnan, P. Chaudhuri, K. Ramasubramanian, and M. Kulkarni, “Improving the learnability of classifiers for sanskrit ocr corrections,” in *The 17th World Sanskrit Conference, Vancouver, Canada*, 2018.