Error Detection in Indic OCRs

V. S. Vinitha and C. V. Jawahar
International Institute of Information Technology, Hyderabad, INDIA

***Abstract*—A good post processing module is an indispensable part of an OCR pipeline. In this paper, we propose a novel method for error detection in Indian language OCR output. Our solution uses a recurrent neural network (RNN) for classification of a word as an error or not. We propose a generic error detection method and demonstrate its effectiveness on four popular Indian languages. We divide the words into their constituent *aksharas* and use their bigram and trigram level information to build a feature representation. In order to train the classifier on incorrect words, we use the mis-recognized words in the output of the OCR. In addition to RNN, we also explore the effectiveness of a generative model such as GMM for our task and demonstrate an improved performance by combining both the approaches. We tested our method on four popular Indian languages and report an average error detection performance above 80%.**

***Keywords*—Error detection, Aksharas, OCR, Indian languages**

I. Introduction and Related Work

An Optical Character Recognition system (OCR) can be used to convert scanned document images to editable electronic documents [1]. Although efforts have been made to build OCRs for Indic scripts, the effectiveness of the techniques employed in English OCRs is not reproducible in Indic scripts. While a lot of emphasis has been placed on building good OCRs, little has been done in the area of post processing in Indian languages. The challenges in the recognition process can be overcome by using a good post processing module. Automating the detection of OCR errors can help improve the overall OCR accuracy in applications where there is a human in the loop. This is also the first step towards automatic error correction.

When a document image is given as input to the OCR it processes the image and outputs the text in an editable form. During this conversion, a single character or a group of characters in the document image can be mis-recognized by the OCR for a similar shaped character or a group of characters, resulting in error. Post processing systems have error detection module to detect such errors and correction module to provide appropriate replacement words. The errors can be broadly classified as real word errors and non-word errors. Real word errors like the word 'aim' getting recognized as 'arm' are difficult to detect without using any context information (like neighbouring words) because both are valid words in the language. However, non-word errors like 'aim' recognized as 'oim' can be detected using character ngram models [2]. Figure 1(a) shows a real word error in Telugu language where a word is incorrectly recognized, and the output is also a valid word. Such errors are difficult to detect without using the context of reference of the word. Figure 1(b) shows an example of a non-word error in Gujarati language, where a group of characters are mis-recognized as another single character.

In this paper, we propose an error detection technique for Indian languages using a combination of Recurrent Neural

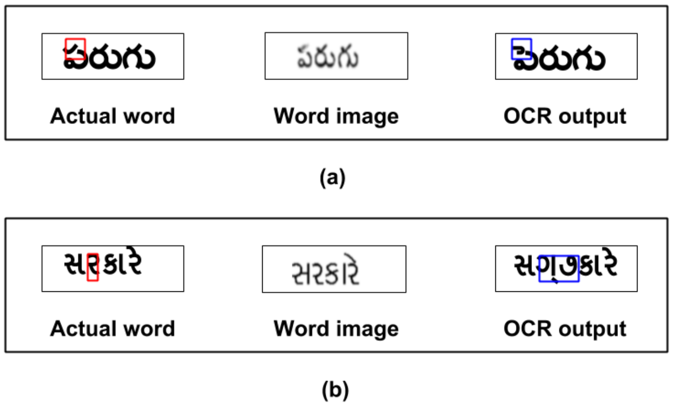


Fig. 1. Types of OCR errors: First image shows a word in Telugu language, its image and the mis-recognized OCR output which is another valid word in Telugu. A unicode character (in red box) is recognized as another unicode character (in blue box). Second figure shows a Gujarati word, the image of which when recognized, gives an invalid word as output. The character (in red box) is recognized as a group of characters (in blue box)

Network (RNN) and a Gaussian Mixture Model (GMM). In Indian languages, words are composed of *aksharas* which are similar to syllables in English. We divide words into their constituent *aksharas* and use their bigram and trigram probabilities to build features for training the classifiers. RNN learns the pattern of word formation using *aksharas* in correct and incorrect words from their bigram and trigram probabilities. We use the GMM model to check the misclassification of correct words as errors by the RNN. The error detection approach essentially requires the learning of patterns which can distinguish a word as error or not. RNN offers good trainability as well as flexibility of using arbitrary input sequence length. Our method can be used on any language without requiring knowledge of the intricacies of its grammar, provided we have a fairly large and clean corpus. Unavailability of a large corpus prompted us to use a web crawler to take advantage of the huge digital content available online.

One can argue that the use of a dictionary is the most straight forward approach to detecting the erroneous words in the OCR output [3, 4]. The presence/absence of a word in the dictionary is used to check the validity of the word. The availability of a nearly complete dictionary can give good results in this method, if one exists! The existence of a large number of unique words is one of the major challenges in Indian languages [2]. To cover around 80% of the words in the language, when English requires around 8K words, Telugu and Malayalam require close to 300K words. Evidently, the main challenge in this approach is the creation of a dictionary which covers most words in the language. Many spell checking and correction systems make use of this binary dictionary method efficiently [5]. The enormous English corpus shared by Google is used to check the validity of words in the OCR output in [4].

Detection of the non-word errors can be done using a dictionary, but capturing real word errors in the OCR output requires the context information. Detection and correction of real word errors is analogous to spell checking in typed text. For the detection of real word errors, a trigram based noisy-channel model is employed in [6]. In noisy-channel model, the goal is to find the intended word given an erroneous word. In some cases, for example, when the error word is a proper noun which is not present in the dictionary, it may be better to accept the error word as the intended word rather than attempting to find an intended word in the dictionary. In [7], the author uses part-of-speech trigrams combined with Bayesian methods for context sensitive spelling correction. In [8], Smith uses a shape classifier model, a word ngram model and a binary ngram dictionary model to detect errors. These methods are effective in detecting errors in English. However in Indian languages, the use of these methods are not effective due to various reasons like complexity of scripts, existence of huge vocabulary etc.

Attempts have been made to use post processors to improve the accuracy of Indian language OCRs [9, 10, 11]. In [9], a multi-stage graph based reasoning, aided by sub-character level language model is used to correct errors in the OCR output. Here unicode characters are used as the basic unit of a word to create the language model. In [10], a shape based post processing system for Gurmukhi OCR was employed in which the Gurmukhi corpora is split into different partitions based on the size and shape of a word. The error detection technique in [11] is done by morphologically parsing the word and checking if the root and suffix part of the word can exist grammatically.

Compared to English and other Latin language OCRs, Indian language OCRs are still inferior [2] mainly because of the following reasons: (a) existence of huge vocabulary (b) longer words (c) large number of valid words within a particular hamming distance (d) inability to perfectly segment characters in a word and (e) lack of language resources like morphological analysers or synthesizers. The high inflectional and agglutinative nature of these languages serve as a major reason for the existence of huge vocabulary. Inflection is the property by which a word is modified to represent different grammatical categories which leads to creation of different words from the same root word. The existence of a huge vocabulary, along with the unavailability of a huge corpus, limits the possibility of using a binary dictionary approach to detect errors. This also affects the creation of a good language model.

The average word length of most Indic scripts are greater than English. Hence for a specific character accuracy, the word accuracy of such scripts would be less. The nature of the script and the complexity of the script further worsens the accuracy in these languages. This also causes issues in perfect character segmentation. Another issue is the number of words in the language which exist at a hamming distance of one or two characters from each other. The percentage of words within a particular hamming distance, is much larger when compared to English [2]. Hence the probability of real word errors is much higher in these languages, making error detection even harder. In the correction of errors, when there are many candidate words for error word replacement, the lowest hamming distance word is chosen. However, when many words with the same hamming distance exist, choosing the right word is a random choice.

Our approach captures up to 87% of the errors in the languages we tested, also ensuring minimum misclassified correct words. The F-score of the method is significantly better than results obtained in [2] which employs a dictionary, statistical language model and a SVM classifier. The catch here is that we do not use any dictionary lookup to predict the word label.

II. Methodology

A. Basic OCR Model and Error Detection Procedure

Generally in OCR systems, each non-touching independent symbol, hereafter referred to as glyph is the basic recognition unit. The segmentation of each word involves identification of these glyphs. Glyphs are segmented using connected components. This is depicted in figure 2. In English, this level of segmentation is relatively easy because of the simple nature of the script. A major issue in Indic scripts is that the glyph level segmentation is overlapping due to the complexity of scripts(note the overlapping boxes). Most Indic scripts have dependent vowels or other modified consonants which are connected to a consonant. These are seen in the red coloured bounding boxes in the figure. Hence perfect glyph segmentation is a challenge. This causes recognition issues leading to errors in the output such as a glyph getting mis-recognized as another glyph or group of glyphs. Post processors are useful in this context wherein, a good language model and knowledge of error patterns can handle the issues created by similar shaped glyphs.

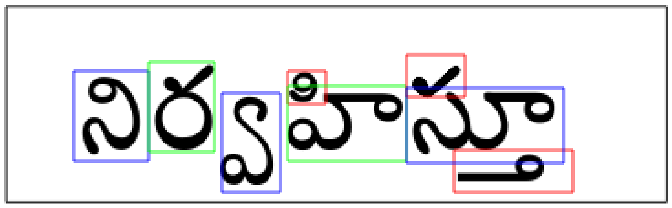


Fig. 2. The figure shows in coloured boxes, the segmentation of each disjoint glyph in a Telugu word.

One of the primary issues in Indian languages is the unavailability of a huge corpus like the British National Corpus [13], Brown Corpus etc. which are available for English. A huge corpus is essential to create a good language model for any language. The unique word coverage of Indian language corpora like CIIL are not sufficient for our application [14]. To include words across domains, we use a corpus made by crawling popular news sites and other websites in Indian languages. The crawled data contains noise due to unwise use for Zero-Width Joiners (ZWJ) and Zero-Width Non-Joiners (ZWNJ) which are used for proper rendering of the unicode symbols. Also many unicode characters which do not belong to the concerned language may also be present in the crawled data. The unicode range for the language is used to filter out the undesirable characters from the corpus. Further, simple cleaning techniques like eliminating words with occurrence of successive vowels are also done.

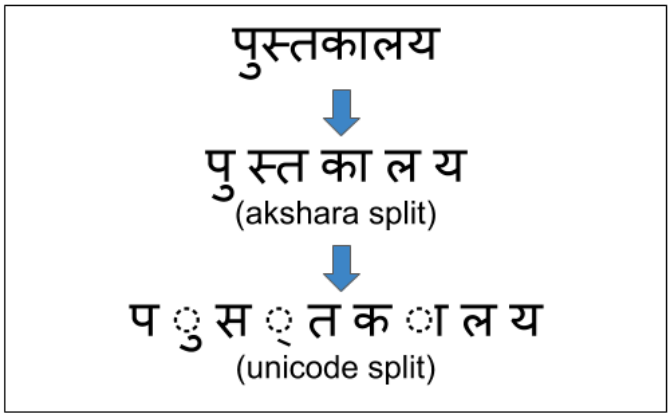


Fig. 3. Figure shows the splitting of a Hindi word at *akshara* and unicode level. An *akshara* may be composed of one or more unicodes.

B. Dilemma: What is the basic Unit?

In Indian languages, the question of choosing the basic recognizable unit of a word is debatable. Each character or glyph has a unicode value associated with it. Decomposition of a word into its constituent unicode characters certainly ensures atomicity, but fails to give insight as to how these bigrams or trigrams can build a meaningful word. This means that the bigram or trigram unicodes may not really say anything about the correctness of a word. This prompts us to go for the *akshara* level splitting of the word. *Aksharas* are similar to syllables in English language, except that in certain languages like Malayalam, a syllable may be composed of more than one *akshara*. We would be using the terms *akshara* and syllable interchangeably. Figure 3 shows a Hindi word split at *akshara* and unicode level. For error detection in OCR output, we find that syllable level splitting better suits our requirement. Word formation is related to morphological as well as phonological features [15]. Syllables provide phonological information and are widely used in speech recognition systems [16]. Splitting of words into *aksharas* can be done using a simple regular expression. *Akshara* is formed using zero or more consonants followed by a vowel. When a word is split into its constituent syllables, if the syllables formed does not belong to the set of syllables already created from the corpus of large words, it is likely that an error has occurred. The words in a particular language has a set of commonly used syllables. This set is not finite, yet if a large corpus is used, we get a fair share of the commonly used syllables. The presence of errors in a word often results in the formation of syllables which are generally not found in the language. However, with the increasing influence and incorporation of words over time from other languages, especially English, the number of syllables in the language is also increasing. For example, many English words like stall, bag, office etc. are widely transliterated to Indian languages, introducing new syllables. Also, in error words, even if the constituent syllables are valid, its bigram or trigram combinations may have less probability. In our experiments, we first split the words into their constituent syllables to compute their bigram and trigram probabilities in the corpus. Table I shows the number of unique words in the crawled corpus, which is used to create syllables in each language, along with the number of unique syllables, bigram and trigram counts.

The SRILM toolkit [17] is used to compute the bigram and trigram probabilities of syllables and smoothing is done using Good-Turing discounting to estimate the probabilities of unseen objects [18]. Probability computations are done for ngrams using nth-order Markov chain assumption and log probabilities are used in computations, since the probability values are very small. The probability of unseen syllable ngrams are taken care of using the smoothing technique as shown below.

$$p_0 = \frac{N_1}{N} \qquad p_r = \frac{(r+1)S(N_{r+1})}{NS(N_r)}$$

where p_0 is the probability for an unseen syllable ngram, p_r is the probability for an ngram encountered r times, N is the total number of ngrams, N_i is the count of ngrams occuring i times and S is a smoothing function. The simple Good-Turing (SGT) method uses a simple linear smoothing function and also specifies a threshold for switching from Good-Turing estimate to Maximum Likelihood Estimate (MLE) for higher frequencies as Good-Turing estimate is accurate only for lower frequencies. We create a lookup table (LT) of these syllables along with their bigram and trigram probabilities for creating features of the words.

TABLE I. STATISTICS OF UNIQUE WORDS AND SYLLABLES IN DIFFERENT INDIAN LANGUAGES.

Language	Unique Words	Unique Syllables	Bigram Count	Trigram Count
Hindi	891,960	15,805	313,989	407,534
Malayalam	398,887	7,257	124,033	176,087
Gujarati	643,986	7,889	172,581	271,075
Telugu	1,305,852	10,762	254,960	441,806

C. Structure of the Solution

We use two methods for detecting errors in the OCR output; one using generative model and the other using a neural network. In generative approach, we use a Gaussian Mixture Model to create models for correct words and error words in the OCR output. In the second approach, we use BLSTM [19] deep learning neural network for classification. In order to create features for training, we have used the bigram and trigram probability of syllables in the corpus, obtained from the lookup table LT. We split each word in the huge corpus into its constituent syllables and add special characters to mark the beginning and end of the word. This is important because in Indian languages, only a specific set of syllables can occur at the beginning of any word. Certain unicode character combinations which occur in the erroneous words, may not be present in the list of syllables created from the huge corpus. We assign a very low probability value to bigrams and trigrams containing these character groups.

D. Gaussian Mixture Model for Error Detection

In this method we first cluster the probabilities of all syllable bigrams in the corpus, using K-means clustering to create bags of syllable bigrams. We have found $K = 10$ to be the optimum number of clusters giving good results by testing on validation data. Each bag has a minimum probability and a maximum probability bigram. We then use this bag of syllable bigrams to create a histogram of each syllable split word. The

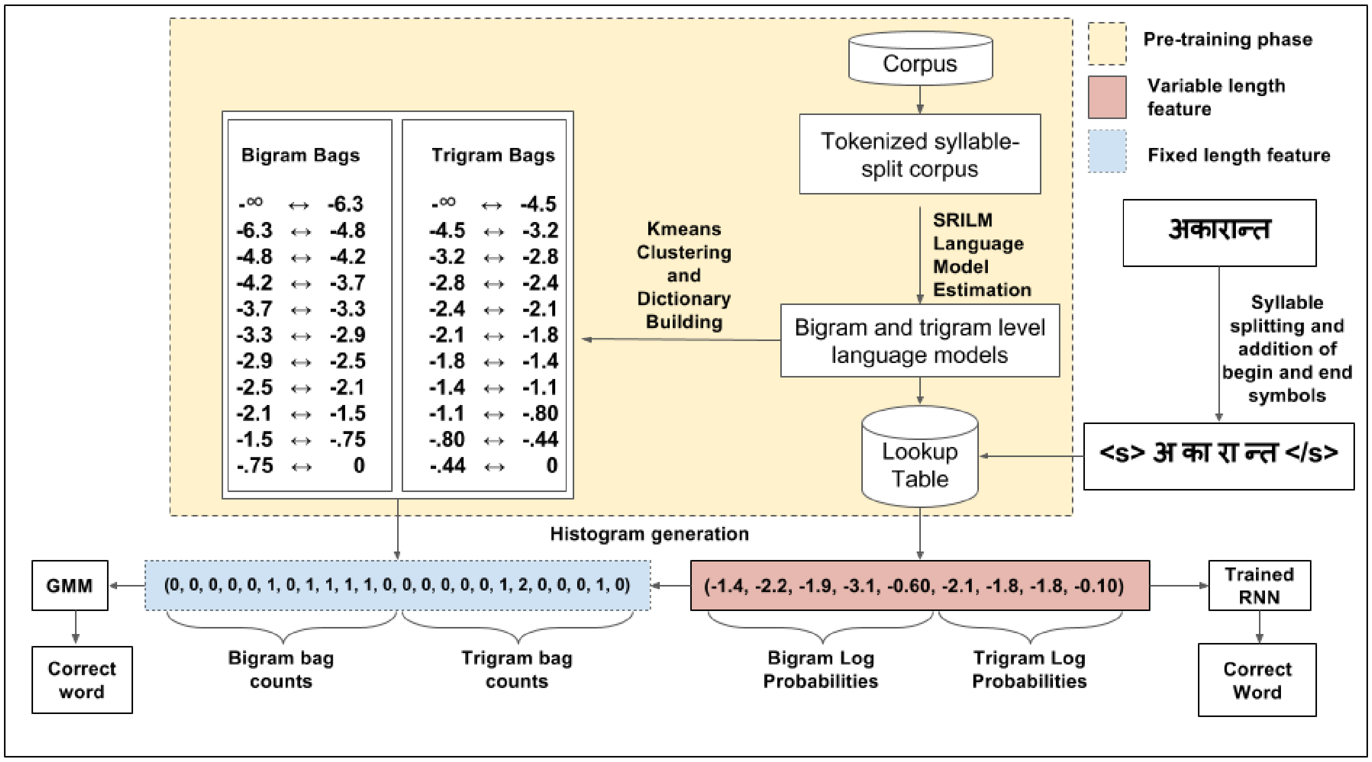


Fig. 4. The image shows how feature is created from a word for RNN and GMM training. After adding markers to the beginning and end of *akshara* split words in a huge corpus, its language model is generated. The bigram and trigram models are clustered separately. We then perform Dictionary Building to find the cluster centroids and create bags of syllables, which is stored as a lookup table. The GMM model takes as input, the fixed length histogram of the syllable split words whereas the RNN uses the raw bigram and trigram syllable probability. Each model then makes a prediction of the label of the input word. A word is declared error only if both the models label it as an error.

same procedure is done for probabilities of syllable trigrams. If there are J bags for syllable bigrams and N bags for syllable trigrams, the size of the feature is $J+N+2$. Here the 2 is for the additional bags for unseen syllable bigrams and trigrams. The steps to creating feature vector for a word are as follows:

1) Create a zero vector of dimension equal to J.
2) For each syllable bigram in the word identify the bag j in which the bigram probability lies.
3) Increment by one, the count of the j^{th} component in the feature vector.
4) In case of new syllables, we increment the count of the bag reserved for unseen bigrams.
5) Repeat the procedure for trigrams using zero vector of size N.
6) Concatenate the above feature vectors to get the final feature vector.

These two histograms of bigram and trigram probabilities are used to create a Gaussian Mixture Model. In the model we used validation dataset to find the optimum number of components so that issues of over fitting of data (with too many components) are taken care of. The procedure is done for obtaining the models for correct words as well as erroneous words. For each word in the testing data, we find the model which best fits the histogram of the word. The word is declared error if it fits the error model and correct otherwise. In GMM model, we use the information in the language model to predict the label of the words. When the GMM is given an unseen word whose syllable bigram and trigram probabilities are comparable to the trained valid word probability, it can use the language model information to correctly predict the label of the word. We preferred to use GMM over other generative methods because of the flexibility it offered in selecting the number of mixture components and its ability to cluster multi dimensional data of unknown distribution.

E. Error Detection using RNN

Recurrent neural network (RNN) is a class of neural networks with the capability of persisting the information from previous states. The loops or connections in the nodes of the recurrent neural network enable it to use an “internal memory” to remember and process past information[20]. In our problem of error detection in OCR output, we use a Long Short Term Memory (LSTM) network. The LSTMs have been used in a wide range of problems including text recognition in images and generating language models. Bidirectional RNNs are based on the idea that the output at particular time may not only be dependent on the previous elements in the sequence, but also on the future elements. We prefer the use of LSTM for error word detection over other classifiers like support vector machines. A neural network can learn the error model in the erroneous words during training. Apart from the advantage offered by the use of networks for better learning, it also provides flexibility of using arbitrary number of sequences as input. The number of unicodes or *aksharas* in words are not fixed, leading to different number of bigrams and trigrams in different length words. We need not use padding or other methods to create fixed length feature while using a LSTM. While GMM uses bags of *akshara* level ngram probabilities, RNN uses the raw values of probabilities for training. For each bigram and trigram in the word, the bigram probabilities, followed by trigram probabilities form the feature vector for the word. The size of the feature for each word having n syllables is $2n - 3$, the sum of the number of bigrams and trigrams in the word. Figure 4 illustrates the feature creation and prediction in GMM and RNN.

III. Experiments

In order to create error words for training, we used the OCR outputs of Hindi, Gujarati, Malayalam and Telugu OCRs [12]. We used 5K document images from each language and used the OCR output collected from the respective OCRs. Recursive Text Alignment Tool (RETA) [21] is used to align the OCR output with the annotated ground truth text and extract the misrecognized words. We have ignored numbers, punctuations, special characters etc. which are not identified correctly by the OCR.

A. Data and Evaluation Metrics

The details of the data used for training and testing using RNN and GMM is shown in table II. We used a train-val-test split ratio of 64-16-20 in the experiments. In order to evaluate

TABLE II. Details of Training and Testing Corpus Size

Language	Words for Training		Words for Testing	
	Errors	Correct	Errors	Correct
Hindi	81,632	89,196	20,308	22,299
Malayalam	966,16	137,171	24,155	34,293
Gujarati	150,825	171,730	37,706	42,932
Telugu	149,501	174,113	37,376	43,529

the error detection accuracy, we use True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) values. TP gives the percentage of errors correctly detected and TN gives the percentage of correct words rightly detected by the post processor. The FN value shows the undetected errors and its value goes up when there are more real word errors. When the correct words are labeled as errors, the FP value increases. This can occur when the correct word pattern is not recognized by the post processor A good error detection system should give significant TP without generating much FP. This means that while all/most of the error words are detected correctly, the percentage of correct words labeled as errors should be kept minimum if not zero. We have used Precision, Recall and F-measure to compare the results of various approaches. Our aim is to maintain a high precision value because large number of correct words recognized as errors make the error detection module insignificant in post processing.

B. Results of using RNN and GMM Methods

The results of error detection experiments using RNN is shown in table III. While Malayalam, Gujarati and Telugu have comparable values of True Postives, many errors went undetected in Hindi. Analyzing the results, we identify the presence of many valid words as errors. This behavior is justified by the presence of large number of words at a particular Hamming distance [2] in Hindi. Therefore, when a character is mis-recognized by the OCR, there is a good chance that another valid word is formed, which is difficult to detect. Other False Positives include words which are not inherently found in the language such as names of people, places etc. The table IV shows the results of both RNN and GMM methods.

TABLE III. TRUE POSITIVE, FALSE POSITIVE, TRUE NEGATIVE AND FALSE NEGATIVE PERCENTAGE FOR LANGUAGES

Language	TP	TN	FP	FN
Hindi	72.30	90.90	9.10	27.70
Malayalam	87.56	94.23	5.77	12.44
Gujarati	83.47	93.70	6.30	16.53
Telugu	80.34	95.69	4.31	19.66

TABLE IV. Comparing Precision, Recall and F-Score values for RNN and GMM approaches. (The values are shown in percentage)

Language	RNN			GMM		
	Precision	Recall	F-score	Precision	Recall	F-score
Hindi	89.30	77.22	82.82	85.46	77.44	81.25
Malayalam	93.82	87.56	90.58	88.47	84.70	86.54
Gujarati	92.98	83.47	87.97	92.05	80.28	85.77
Telugu	94.91	80.34	87.01	92.44	79.55	85.51

While comparing the F-measure values of both the approaches, we can see that RNN based approach performs better than GMM. This can be attributed to the effective learning capability of neural networks. It is also observed that the Recall of Hindi is almost same in both approaches. The effectiveness of both the approaches can be combined to build a powerful post processor.

C. Combining RNN and GMM Approaches

One of the important concerns in OCR post processing is the misclassification of correct words identified correctly by the OCR. The cost of misclassifying a correct word in the OCR output as 'error' by the post processor is much higher than the cost of not identifying an error. This implies that we should be more concerned about increasing the precision. A good post processor should try to minimize the occurrence of False Positives while also trying to maximize the True Positives.

As observed in [8] relying on one method can fix some obvious errors but it can also increase the rate of hallucination of correct words as errors. We combine both our approaches to create a more reliable classifier wherein a word is declared as an error only if both the models label it as an error. The table V shows how a word is given a label from the labels of RNN and GMM approach. The results of the combined approach for different languages are shown in figure 5.

TABLE V. Rules for labeling a word by combining the models

RNN output	GMM output	Combined Approach Output
Error	Error	Error
Error	Right	Right
Right	Error	Right
Right	Right	Right

D. Discussions

The error detection in OCR output using RNN and GMM gives us good detection accuracies. The primary reason for this is the exploitation of the potential of a neural network and complementing its predictions using a generative method. Also the use of *akshara* as the basic recognition unit of a word helps

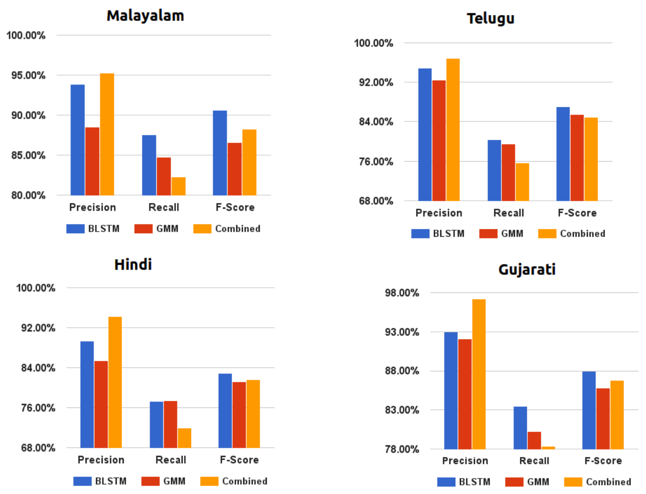


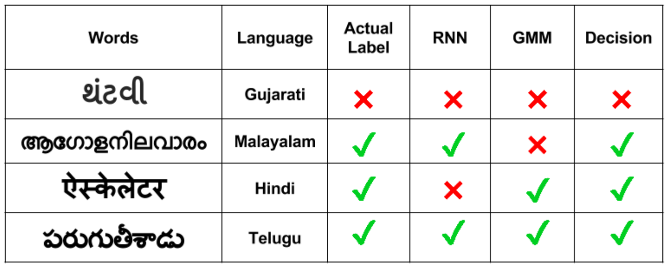
Fig. 5. The bar graph shows the precision recall and F-score using RNN, GMM and the combined approach. The precision in the combined approach exceeds both the individual approaches. We compromise recall to improve precision, because our problem demands that the misclassification of correct words should be minimized.

TABLE VI. Results of error detection on a new RNN based OCR using the pre-trained error detection model. We have used a combination approach discussed in previous session to achieve this result. The values are in percentage.

Language	TP	TN	FP	FN	Precision	Recall	F-Score
Malayalam	75.57	85.38	14.62	24.43	83.79	75.57	79.47
Gujrathi	67.39	94.83	5.17	32.61	92.87	67.39	78.11
Hindi	64.83	87.98	12.02	35.17	84.36	64.83	73.32

in learning the morphology of a word and patterns in word formation, enabling better prediction of labels of unseen words. The method fails to detect correct words like person names or place names which are not related to the region where the language is used. Also detection of errors in punctuation and digits is a troublesome task. Overall, the approach succeeds in providing a fair solution for detecting non word errors in the OCR output.

We evaluate the error detection accuracy on the output obtained from another OCR (RNN based) and the results are shown in table VI. Some qualitative results of the combined approach are shown in figure 6. We observe that long words like the Malayalam word, which are actually correct are identified correctly by the RNN. The GMM does well at picking up transliterated words from languages like English as shown in the Hindi example. It can be seen that combining both the models helps to reduce misclassification of correct words.



Words	Language	Actual Label	RNN	GMM	Decision
થંટવી	Gujarati	✗	✗	✗	✗
ആഗോളനിലവാരം	Malayalam	✓	✓	✗	✓
ऐस्केलेटर	Hindi	✓	✗	✓	✓
పరుగుతీశాడు	Telugu	✓	✓	✓	✓

Fig. 6. Figure shows some of the test cases and the labels assigned to them by each model. Cross mark and tick mark indicates that the label is error and correct respectively. Decision column shows the prediction made by combined method.

IV. Conclusion and Future Work

Detection of errors in the output of the OCR is an inevitable task in making good OCRs, especially for Indic scripts. In this work, we provide an effective solution to detecting the errors in the OCR output using a pretrained LSTM and GMM model. We use the bigram and trigram probabilities of *aksharas* in a word to train the models. In the future, we would like to try new features to train the neural network and also use word level ngram features to predict the real word errors in the OCR output. We also plan to extend this work to other language OCRs.

References

[1] M. Cheriet, N. Kharma, C. L. Liu, and C. Y. Suen, *Character Recognition Systems*. Wiley-Blackwell, 2007.
[2] N. Sankaran and C. V. Jawahar, "Error detection in highly inflectional languages," in *ICDAR*, 2013.
[3] K. Kukich, "Techniques for automatically correcting words in text," *ACM*, 1992.
[4] Y. Bassil and M. Alwani, "OCR context-sensitive error correction based on google web 1T 5-gram data set," *American Journal of Scientific Research*, 2012.
[5] A. Carlson and I. Fette, "Memory-based context-sensitive spelling correction at web scale," in *Machine Learning and Applications*, 2007.
[6] A. Wilcox-OHearn, G. Hirst, and A. Budanitsky, "Real-word spelling correction with trigrams: A reconsideration of the Mays, Damerau, and Mercer model," 2008.
[7] R. Golding and Y. Schabes, "Combining trigram-based and feature-based methods for context-sensitive spelling correction," in *ACL*, 1996.
[8] R. Smith, "Limits on the application of frequency-based language models to ocr," in *ICDAR*, 2011.
[9] K. Mohan and C. V. Jawahar, "A post-processing scheme for malayalam using statistical sub-character language models," in *DAS*, 2010.
[10] G. Lehal, C. Singh, and R. Lehal, "A shape based post processor for Gurmukhi OCR," in *Document Analysis and Recognition*, 2001.
[11] 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 Engineering*, 2000.
[12] D. Arya, T. Patnaik, S. Chaudhury, C. V. Jawahar, B.B.Chaudhuri, A.G.Ramakrishna, C. Bhagvati, and G. S. Lehal, "Experiences of integration and performance testing of multilingual OCR for printed indian scripts," in *J-MOCR Workshop,ICDAR*, 2011.
[13] British National Corpus (BNC). [Online]. Available: http://www.natcorp.ox.ac.uk/
[14] A. Bharati, K. Prakash Rao, R. Sangal, and S. Bendre, "Basic statistical analysis of corpus and cross comparison among corpora," *Technical Report of Indian Institute of Information Technology*, 2000.
[15] Y.-S. Hwang, B.-R. Park, H.-C. Rim, and S.-W. Lee, "A contextual post-processing model for Korean OCR using synthesized statistical information," in *ICMI*.
[16] A. Ganapathiraju, J. Hamaker, J. Picone, M. Ordowski, and G. R. Doddington, "Syllable-based large vocabulary continuous speech recognition," *Speech and Audio Processing*, 2001.
[17] A. Stolcke *et al.*, "Srilm-an extensible language modeling toolkit." in *INTERSPEECH*, 2002.
[18] W. Gale and G. Sampson, "Good-turing smoothing without tears," *Journal of Quantitative Linguistics*, 1995.
[19] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition," *Pattern Analysis and Machine Intelligence*, 2009.
[20] L. Medsker and L. Jain, "Recurrent neural networks," *Design and Applications*, 2001.
[21] I. Z. Yalniz and R. Manmatha, "A fast alignment scheme for automatic ocr evaluation of books," in *ICDAR*, 2011.