Recognition of Printed Devanagari Text Using BLSTM Neural Network

Naveen Sankaran and C.V Jawahar
International Institute of Information Technology - Hyderabad, India

Abstract

In this paper, we propose a recognition scheme for the Indian script of Devanagari. Recognition accuracy of Devanagari script is not yet comparable to its Roman counterparts. This is mainly due to the complexity of the script, writing style etc. Our solution uses a Recurrent Neural Network known as Bidirectional Long-Short Term Memory (BLSTM). Our approach does not require word to character segmentation, which is one of the most common reason for high word error rate. We report a reduction of more than 20% in word error rate and over 9% reduction in character error rate while comparing with the best available OCR system.

Keywords: BLSTM, Word recognition, Devanagari, OCR

1. Introduction

Most of the present day Optical Character Recognizers (OCR) show impressive results for a wide range of documents in Roman scripts. Past few years have seen considerable interest in developing similar OCR systems for Indic scripts [6]. Several improvements have been made in this frontier which resulted in significant advancement in techniques and improvements in results [1, 6]. Still the accuracies drop heavily with degradations.

Most of the traditional OCRs use character segmentation to extract symbols and recognize them using a classifier. However, such methods fail in poor quality documents due to the presence of cuts and merges. Degradations can arise due to multiple sources like poor quality of ink, low spacing between characters, document age etc. Considerable effort was made to find a solution to this problem at character level [10]. However, trying to find a solution at character level may not be the ideal approach, as such methods may fail when it comes to large diverse documents containing variation in font, degradation etc.

We believe that recognition of degraded characters requires more than just the knowledge of which connected components can be joined/split to form a valid component. Using contextual information can be a good way to solve such problems. A word, when looked as a single component, rather than a collection of characters, provide more information than what an isolated character can provide. Recognition at word level is not new. There have been several other word recognition systems in the past which tried to address this issue [11, 3, 9]. We propose to recognize Devanagari words by using a neural network classifier known as Bidirectional Long Short-Term Memory (BLSTM) neural network [7]. BLSTM based word recognition have been proven quite successful in recognizing a variety of documents including handwritten documents [4]. We have used the same neural network for word retrieval for Hindi documents [8]. In this work, we extend this towards recognition.

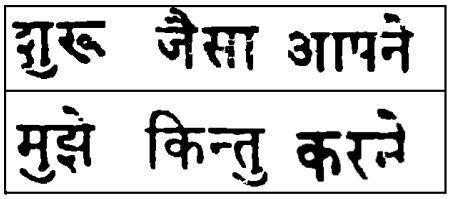

Figure 1. Some word examples which are being correctly identified by our method. Presence of degradation prevent traditional OCR to recognize them correctly.

We also used a script-dependent segmentation module to obtain the word images from pages. This was needed largely due to the presence of complex, multi-column pages. Our Devanagari-specific layout module took page images as input and generated a set of word images.

2. Word Recognition using BLSTM

We recognize Devanagari text based on a variant of Recurrent Neural Networks, known as Bidirectional Long Short-Term Memory (BLSTM) neural network [7]. The BLSTM consists of hidden layers which are made up of the so-called long short-term memory (LSTM). Such a solution enables the network to predict

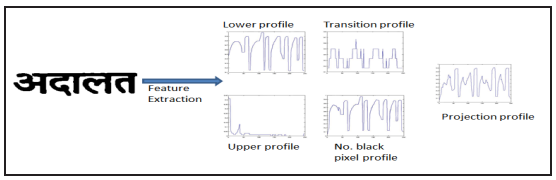

Figure 2. A sample word image and the five features extracted from it.

the label sequence based on both the past and future context of the element. This is done by making use of two hidden layers. One to process the input sequence forward while other processes it backwards. Similar system has been used in word recognition and retrieval in the past [8]. Some of the advantages of the system are mentioned in [12] [5]. The data is recognized at word level instead of character level so as to overcome the issues associated with degraded data while taking into account the context associated within a word.

The problem of vanishing gradient, i.e error gradients vanish exponentially with the size of the time lag between important events, has been addressed in the BLSTM network by using the LSTM hidden layers. The LSTM hidden layers consist of recurrently connected subnets, called memory blocks. Each block consists of a set of internal units whose activation is controlled by three multiplicative gates: the input gate, forget gate and output gate. Information can be stored and accessed over a long period of time using these gates. More details are present in [7].

The words written in Devanagari script are connected by a head line, known as *shirorekha*. Vowel modifiers or *matras* can be made by combining a vowel with a consonant. The script also allows joining of multiple vowels and consonants to form a compound character. This results in having the number of unique characters anywhere between 300-800. This number depends on the font and the rendering scheme used. To reduce the number of unique classes, a commonly followed approach is to remove the *shirorekha* and recognize the characters and then recombine those at the end. This is referred to as zoning [2]. Our system is robust enough to handle the large number of classes and hence, we do not perform zoning of words. In this regard, our present work is considerably different from [8]. The number of unique class labels stand at 685 in our case.

For every word, we extract 5 different features from a vertical strip of uniform width, using a sliding window. The features extracted are (a) the lower profile, (b) the upper profile, (c) the ink-background transitions, (d) the number of black pixels, and (e) the span of the foreground pixels. The upper and lower profiles measure the distance of the top and bottom foreground pixel from the respective baselines. Ink-background transitions measures the number of transitions from Ink to background and reverse. the number of black pixels provides the information about the density of ink in the vertical stripe. A sample word and its corresponding feature sequences are shown in Figure 2.

We represent the word image as a sequence of feature vectors of dimension d. BLSTM neural networks contain one node for each of the features in the input layer. Thus the network has d input layer nodes. Each node in the input layer is connected with two separate hidden layers, one of which processes the input sequence of features forward, while the other processes it backward. Both hidden layers are connected to the same output layer. The output layer sums up the values which comes from both forward and backward hidden layers. Most of the Recurrent Neural Networks(RNN) require pre-segmented input label with a separate target for every label. However, such a mechanism might prove to be faulty in cases where segmentation would be tough. An output layer known as Connectionist Temporal Classification(CTC) is used to overcome this issue. This kind of layer has got the ability to directly output the probability distribution over label sequences. We normalize the output activation functions in such a way that the result is one when they are summed up. This is then treated as probability vector of the characters present at that position. The output layer contains one node for each class label plus a special node ϵ , to indicate "no character",i.e: no decision about a character can be made at that position. Thus, there are $K + 1$ nodes in the output layer, where K is the number of classes.

The CTC objective function is defined as the negative log probability of the network correctly labelling the entire training set. For a given training set (S) consisting of pairs of input and target sequences (x, z), the objective function O can be expressed as

$$O = -\sum_{(x,z)\in S} \ln p(z|x).$$

One advantage of having such a discriminative objective function is that we can directly model the probability of the label sequence given the inputs. This has been proven better than an HMM based methods which are generative in nature [9]. Also an HMM based system assume that the probability of each observation depends only on the current state. On the other hand, a BLSTM system can easily model continues trajectories and the amount of contextual information available to such a network can in principle extend the entire input sequence.

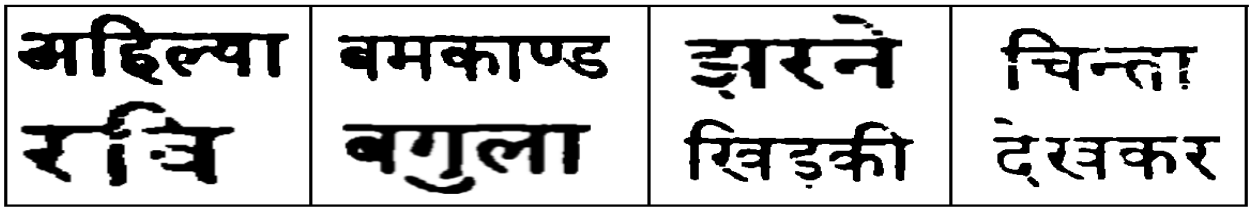

Figure 3. Some examples of words which were correctly recognized by our method. As shown, in-spite of having considerable degradations, we are able to recognize the words correctly.

	CER		WER	
Devanagari	OCR [1]	Ours	OCR [1]	Ours
Good	7.63	5.65	17.88	8.62
Poor	20.11	15.13	43.15	22.15

Table 1. Character error rate(CER) and Word error rates(WER) for Devanagari dataset. BLSTM based method is compared against a traditional OCR system. BLSTM based system shows considerable improvement in accuracy.

3. Results and Discussions

In this section, we quantitatively compare our method against a state-of-the-art Indian language OCR [1]. We perform experiments on two datasets, the details are shown below. The total time taken to test a word image is approximately 0.25 seconds. This includes the time taken to compute features and to classify them.

Figure 4. The first image is subjected to increasing degree of degradations till our system failed to recognize the word correctly.

Dataset: We obtain the word images needed for our experiments from multiple annotated Devanagari books. The groundtruth is obtained from manual typing. The training dataset consists of nearly 90K word images. The validation dataset needed by neural network consists of nearly 74K words and we used around 67K words for testing. The dataset was further split into a Good and Poor datasets, based on the amount of degradation it has. Good dataset consists of words which have considerably less degradation. These are data on which a state-of-the-art OCR performs considerably well. However, the poor dataset contains words with high percentage of degraded words and a traditional OCR system fails badly for these words.

We test our method and compare the results against a state-of-the-art Indian Language OCR [1]. Some of the qualitative results are shown in Figure 5. The tick(green) marks indicate the word was recognized correctly while the cross(red) mark indicates that the word was recognized wrongly. For these examples, OCR is producing wrong outputs because of the cuts present in word image. The red circle shows the area where cut is present. The cut has resulted in converting a single valid character into another set of valid components, due to which the OCR fails.

We then proceeds to stretch our method to know how much degradation it can actually handle. For this, we took a word which was correctly recognized by our method, and introduced synthetic degradation to it. After every step of degradation, the word was again sent back to our method for recognition. The degradations were introduced till our method failed to correctly recognize the word. Some results are shown in Figure 4.

The quantitative evaluation of the entire dataset is done by comparing the labels returned from the neural network against the ground truth data to compute both Character Error Rate (CER) and Word Error Rate(WER). Our results are compared against current Devanagari OCR system [1]. The results are provided in Table 1. As shown in the table, we show considerable improvements in accuracy for good as well as bad dataset. The word accuracy has improved by more than 9% in case of good data and the improvement is more than 20% when it comes to poor data. The higher improvement in accuracy for bad data can be attributed to the fact that it is these data that require better understanding of context to be recognized correctly which is exactly what is being delivered by our method.

Some of the words which we recognized correctly is shown in Figure 3. As shown, we are able to recognize the words correctly even in the presence of multiple cuts or ink spread causing the character to become like a blob etc. Generally, it was observed that the method was able to correctly recognize word labels even if there are cuts or font variations.

Error Analysis. Some examples where our method fails is shown in Figure 6. The primary cause of failure is when there is not enough scope to extract any meaningful information out of the word images. In the Figure 6, example (a) and (b) are heavily degraded due to which any meaningful extraction of information is very tough. Another reason for failure is the confusion which arises due to similarity in characters. In example (c), the small dot present on top of word is actually a valid one. However, due to small size, it was considered as a noise. Moreover, the 'ga' symbol got confused with the '|' symbol. This is also in part due to some amount of erosion which occurred in the word image. Similarly, in example (d), 'na'(second character in image) was confused as 'ta'. This happened because the 'na' character in devanagari has a hole in beginning, which got filled up, resulting in it getting confused with 'ta'. We believe that using a more descriptive features will be able to handle most of the failure cases.

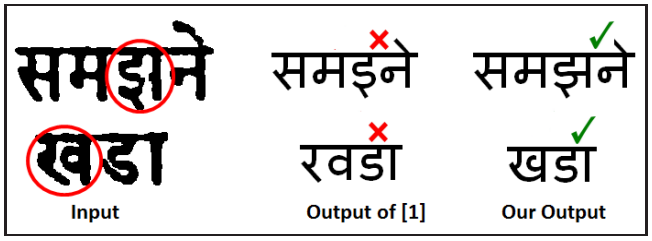

Figure 5. Table showing the OCR output and our output for an image. The reason for OCR error is marked using the red circle.

4. Conclusions and Future Work

In this paper, we proposed a BLSTM based system that performs recognition at word level. It results in more than 20% improvement in word accuracy while comparing traditional OCR system.

In future, we would like to evaluate different features for this network and also use a language model/dictionary based post-processor to improve the accuracy. We would also like to extend our work towards recognition of other Indian languages.

5. Acknowledgment

We would like to thank R. Manmatha, Volkmar Frinken and Raman Jain for introducing us to the BLSTM neural network.

We thank Alex Graves for providing us with an implementation of the neural network. This work is supported by MCIT, Govt. of India.

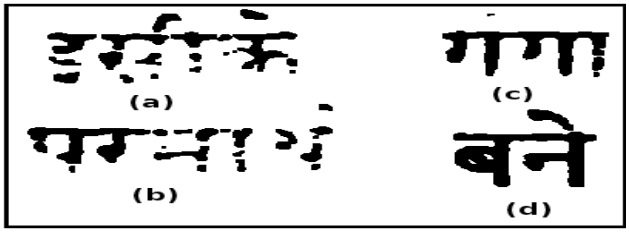

Figure 6. Failure case examples of our methods due to heavy degradation or similarity in characters.

References

[1] 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.

[2] B. B. Chaudhuri and U. Pal. An OCR system to read two indian language scripts: Bangla and Devnagari (Hindi). In *ICDAR*, 1997.

[3] S. Dutta, N. Sankaran, K. P. Sankar, and C. V. Jawahar. Robust recognition of degraded documents using character n-grams. In *Document Analysis Systems*, 2012.

[4] V. Frinken, A. Fischer, and H. Bunke. A novel word spotting algorithm using bidirectional long short-term memory neural networks. In *ANNPR*, 2010.

[5] V. Frinken, A. Fischer, R. Manmatha, and H. Bunke. A novel word spotting method based on recurrent neural networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(2), 2012.

[6] V. Govindaraju and S. Setlur. *Guide to OCR for Indic Scripts*. 2009.

[7] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(5), 2009.

[8] R. Jain, V. Frinken, C. V. Jawahar, and R. Manmatha. BLSTM neural network based word retrieval for Hindi documents. In *ICDAR*, 2011.

[9] P. S. Natarajan, E. MacRostie, and M. Decerbo. The bbn byblos hindi ocr system. In *DRR*, 2005.

[10] U. Pal and B. B. Chaudhuri. Indian script character recognition: a survey. *Pattern Recognition*, 2004.

[11] K. P. Sankar, C. V. Jawahar, and R. Manmatha. Nearest neighbor based collection OCR. In *Document Analysis Systems*, 2010.

[12] H. B. V. Frinken, A. Fischer and R. Manmatha. Adapting blstm neural network based keyword spotting trained on modern data to historical documents. In *Proc. ICFHR*, 2010.