Towards a Robust OCR System for Indic Scripts

by

Praveen Krishnan, Naveen Sankaran T, Ajeet Kumar Singh, C V Jawahar

in

DAS 2014
(*International Workshop on Document Analysis Systems*)

Report No: IIIT/TR/2014/-1

Centre for Visual Information Technology
International Institute of Information Technology
Hyderabad - 500 032, INDIA
April 2014

Towards a Robust OCR System for Indic Scripts

Praveen Krishnan, Naveen Sankaran, Ajeet Kumar Singh, C. V. Jawahar
Center for Visual Information Technology, IIIT Hyderabad, India.

***Abstract*—The current Optical Character Recognition (OCR) systems for Indic scripts are not robust enough for recognizing arbitrary collection of printed documents. Reasons for this limitation includes the lack of resources (e.g. not enough examples with natural variations, lack of documentation available about the possible font/style variations) and the architecture which necessitates hard segmentation of word images followed by an isolated symbol recognition. Variations among scripts, latent symbol to UNICODE conversion rules, non-standard fonts/styles and large degradations are some of the major reasons for the unavailability of robust solutions. In this paper, we propose a web based OCR system which (i) follows a unified architecture for seven Indian languages, (ii) is robust against popular degradations, (iii) follows a segmentation free approach, (iv) addresses the UNICODE re-ordering issues, and (v) can enable continuous learning with user inputs and feedbacks. Our system is designed to aid the continuous learning while being usable i.e., we capture the user inputs (say example images) for further improving the OCRs. We use the popular BLSTM based transcription scheme to achieve our target. This also enables incremental training and refinement in a seamless manner. We report superior accuracy rates in comparison with the available OCRs for the seven Indian languages.**

Keywords*—*Optical Character Recognition, Neural Networks, Indic Scripts

I. INTRODUCTION

Optical character recognition systems for Indic scripts have not yet reached a state, where it is usable for a common man in a robust way for most of the languages. There has been significant progress in the recent past on developing robust solutions [1], [2], [3]. In addition to many technical challenges associated with these scripts, languages and algorithms, this was also partly due to the lack of research manpower (in both academia and industry) that could focus on a script, and engineer a robust system. Most of the previous attempts have been script specific and required high amount of expertise in transferring the knowledge across the scripts. Extensive use of machine learning (not just the design of classifiers for isolated character segments) have helped the document image analysis in multiple ways in the recent years [2], [4], [5], [6], [7], [8]. For scripts/languages, where the research manpower (and also possibly the commercial interests) are lacking, this has greatly helped in rapidly building reasonably robust systems.

While designing the OCRs, the critical learning task is to output a string or word for a given word image. Traditionally this module has been formulated as an *adhoc* composition of a set of isolated character (or symbol) classifiers [1], [9]. Advances in machine learning has also enabled the output prediction as structured prediction task rather than predicting individual class labels and then combining them to obtain semantically richer structures such as a list of class labels or a list of UNICODEs. (Note that for Indic scripts often the classes are defined very differently from the codes used for the representation.) This paper builds on some of our recent attempts [3], [7], [10] to design a generic framework, which is also applicable for multiple Indian scripts.

Earlier attempts on Indian scripts were primarily based on handcrafted architectures which extensively used the script and language specific information. This, while throwing light on the unstructured (rather unmoderated) design and propagation of fonts and styles within the community, also resulted in systems that require significant amount of hand tuning. Because of this it is often susceptible to failure in presence of wild/natural inputs (e.g. a new style/character, extreme degradation etc.). Many of the early solutions were based on rule-based systems with heuristic features [9]. During the last decade or so, there have been significant efforts in collecting and annotating data for Indic scripts [2], [11], [12]. This has resulted in a class of algorithms that employs modern machine learning based solutions (eg. SVMs, CNNs) for classifying the isolated characters. However, converting the classifier outputs built from isolated characters to a structured output (like a string) required many additional steps, and it was seldom taken into the consideration while designing these classifiers. Hidden Markov Models (HMMs) have been extensively used for predicting the words in the context of handwriting recognition. HMM based solutions for printed data [13] also required annotation of data at isolated character level and some extra effort (say a set of rules) that convert the class label sequences to a string of UNICODEs. However, Indian scripts show a larger dependency (not just Markovian) and often non-causal dependency (bidirectional). This has resulted in limited success of these methods. Adaptation of the popular open source OCR solutions to Indic scripts was also adversely affected by the lack of appropriately annotated datasets.

This paper proposes a step towards building robust recognizers for a set of Indic scripts. We also present a web-based system which can robustly recognize a large class of Indian language documents. This covers seven major scripts, and we demonstrate the utility on popular printed content. Our objective is to eventually extend this for a wider styles of documents. However, this requires additional annotations and resources to adapt to other print styles. (Scripts like Malayalam have gone through formal revision of script in 1970s and informal changes with the popularity of computer aided typesetting in late 1980s. Many other scripts like Devanagari have modified certain specific characters during the last three decades. One could still see unexpected combinations even in modern printed documents. The lack of enough such diversities limit our present system.)

Though there have been some amount of annotated data, the quantity and quality of the data is still a challenge. Our web based implementation enables the users to recognize the documents in two modes: (i) interactive, as well as (ii) an API. By providing an interactive interface for recognition, user

is enabled to recognize documents without worrying too much about the complexity of layout and script specific segmentation issues [14]. Web based interface enables the user to refine the pre-processing, and obtain the recognized output to download or receive as an email. Such configurations are for people who work on isolated pages. For those who wish to use OCR as part of a larger application, we also implemented an API, which can be directly used over the web. In both the settings isolated and bulk OCR options are provided. Our web based system is designed to continuously improve the performance over sessions. We use the data provided by the user for improving the performance of the recognizer. (Details of which are not part of this paper.) The internal recognition scheme is also ideally suited for an incremental learning.

This paper is organized as follows. Section II describes a BLSTM based transcription formulation for Indian language OCRs. This is followed by the explanation of our web-based OCR system in Section III. Section IV details the performance of our OCR for seven Indic scripts and our observations on the results.

II. OCR: A Transcription Framework

OCR systems often have multiple intermediate steps while they convert a word image to its corresponding text/UNICODE representation. Word to symbol/character separation, is required for classifiers that recognize isolated characters [1]. Holistic word recognition systems [13] however bypass this step and directly extract features at word level. The extracted features are then sent to the classifier for recognition. Once the classification is done, a set of rules are required for the conversion of latent symbols generated by the classifier to the corresponding UNICODE representation. The rules become a necessity in case of Indic scripts where (i) a symbol could be made using multiple UNICODEs and (ii) the UNICODEs might require re-arrangement so that the word could be rendered properly. i.e. the order of symbols occurring in the image space and the order of those symbols in the text space may not be same.

Developing an OCR engine for a new language is challenging if one is not aware of the set of UNICODE rules. More over, these conversions are often brittle and can fail with noisy symbol labels. In our previous work [7], we have proposed a solution to this problem by considering it as a sequence to sequence transcription, where we convert the sequence of word features into the corresponding text sequence. Such an approach does not mandate one to identify the UNICODE rearranging rules in advance. The system should be capable of learning these rules. While [7] demonstrated the system which recognizes Hindi (a language with relatively fewer rearranging rules), the present work shows the result on multiple Indic scripts, having much complicated rearranging rules. To accommodate such rules it is important for the classifier to analyze the feature segment based on its forward and backward information. We have observed that the BLSTM network, with its unique forward and backward LSTM networks, is well suited for such a transcription task.

A. Feature Extraction

Profile features are among the most popular features which can be used to represent the word images as a feature sequence.

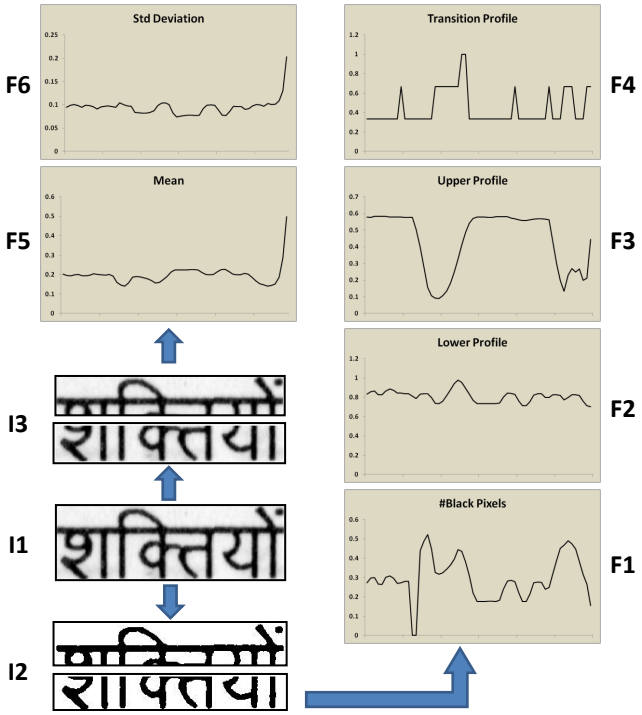

Fig. 1. Figure shows the various features extracted from the upper split of the word image. Here I1, I2 and I3 corresponds to the original gray, split gray and split binary image respectively. F1 to F6 are the features which corresponds to #black pixels, lower profile, upper profile, transition profile, mean and standard deviation respectively. Similar features are computed for both the top half as well as the bottom half for every word image resulting in a total of 12 features for every word.

In this work, we use the split profile features where we divide the image horizontally into two parts and the following features are computed. (i) vertical profile (F1) i.e the number of ink pixels in each column (ii) location of lowermost ink pixel (F2), (ii) location of uppermost ink pixel (F3) and (iv) number of ink to background transitions (F4). The profile features are calculated on binarized word images obtained using the Otsu thresholding algorithm. Apart from these features, we have also used gray level information such as the mean pixel value and the standard deviation among the gray pixel values. Figure 1 shows an example of these features on the word image shown as I1. We compute the features on a one dimensional horizontal sliding window of size of 20 pixels with an overlap of 75%. The features are normalized with respect to the height of the word image. No *shiro-rekha* removal is performed for Devanagari based scripts.

B. BLSTM Networks

Our preferred solution to the transcription problem is a recurrent neural network such as Bidirectional Long-Short Term Memory (BLSTM) network. BLSTM networks have been successfully used in the past for both printed and handwritten text recognition tasks. The distinctive feature about these networks is their ability to remember long range context over several timesteps. The use of Connectionist Temporal Classification (CTC) layer as the output layer allows the words to be presented as a sequence of unsegmented features. This helps to a large extend to reduce the character segmentation

issues in Indian languages. Further the bidirectional nature of the network makes it suitable for UNICODE level learning as it is capable of handling UNICODE reordering issues which are quite prevalent in Indian scripts. These features make BLSTM a natural choice for developing a document image translation system for Indian languages.

A BLSTM network contains two LSTM networks in which one network takes the input from beginning to end while other network takes the input from end to beginning. The individual output of both these networks are used to predict the final output. The final ground-truth alignment is done using a Connectionist Temporal Classification (CTC), which internally uses a forward-backward algorithm. The presence of CTC layer also avoids the requirement of segmenting training data into symbols. 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 labeling 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:-

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

More details into BLSTM network and CTC layer can be found in [15], [16].

We analyzed the network performance on various parameter settings for the Indic OCR problems. A BLSTM network is mainly characterized by the number of hidden nodes (LSTM Size) it uses, number of hidden layers and the stopping criteria used for training. The time complexity for each epoch increases with increasing the LSTM size or the number of hidden layers. We stop the network training when successive training error rates cease to reduce below a certain threshold. We found that increasing number of hidden layers until 3 gave better results. The network with larger LSTM size resulted in poor generalization. The best results are obtained from the LSTM size 50 with 3 hidden layers.

III. Web Based System

We also provide users with a way to evaluate our system via web. We provide a simple web-interface where users can upload scanned images and download the corresponding OCR text. A set of commonly used pre-processing libraries have been utilized to perform tasks like skew correction, word segmentation etc. We have exposed our OCR as web services and any developer who wishes to use this can do so by requesting the *Web Based Services* over Internet for text recognition processes.

User can upload the document images individually or in bulk onto the server. We then proceed to perform pre-processing on the uploaded document images and send the processed images to the recognition engine. After the text extraction has been done, the extracted output will be available for download to the user. We are also proposing to provide authentication services to authenticate the credible users of the interactive and web based services. The documents uploaded by different users help us in improving our systems further. We believe our web-based system will help the development of robust OCRs in the following ways:

- Evaluation of our system by testing it against different fonts, scan quality, page layout etc.
- A self learning system which can adapt to different fonts/styles automatically over time.
- A collection of documents which can be shared across document image community.
- An interface which can be used by developers to build several applications without getting into the specifics of script and algorithms.

IV. Results & Discussion

A separate neural network is trained for each language. For each language, the initialization parameters have been found out experimentally. Number of input nodes is equal to number of input features (12 in our case) and number of output nodes is same as the number of target labels. As mentioned in the previous sections, in order to show the generality of the proposed method, we have taken seven different scripts from Indian languages. Table II shows the details of the dataset used for the experimentation. In each language we have taken around 1000 pages from the corpus which has emerged as a common benchmark data [2] within Indian research community. Almost all of these scripts have its own unique way of representing the character symbols. The scripts of the languages such as Hindi, Bangla and Gurumukhi uses *shirorekha* (headline) over its words while the languages such as Malayalam, Tamil, Telugu and Kannada are more curved in nature. Also the presence of *matras*, *samyuktakshars* and multiple representation of same symbol further makes the script complex. It is to be specially noted that many of the Indian languages are highly inflectional which induces a large vocabulary. The training of the network is done using 60% of

TABLE II. Dataset Details

Language	No. of Books	No. of Pages	No. of Words
Hindi	7	1146	390K
Malayalam	5	1050	210K
Kannada	7	1086	210K
Tamil	8	1052	130K
Telugu	6	1047	200K
Gurumukhi	7	1024	320K
Bengali	6	932	260K

word images and tested on the remaining 40%. The error rates obtained for each language is shown in Table I. The results are computed on independent word recognition outputs. We also compare the results of our method against state of the art OCRs available for these languages. Our method clearly outperforms in terms of both character and word level error rates. The performance trend across the languages are consistent. This indirectly points that such an architecture would be fairly

TABLE I. CHARACTER AND WORD ACCURACY FOR DIFFERENT LANGUAGE CORPUS. WE COMPARE OUR RESULTS AGAINST OTHER STATE-OF-THE-ART OCR SYSTEMS. NOTE THE PROPOSED METHOD DOES NOT USE ANY KIND OF LANGUAGE MODELS OR POST-PROCESSING TECHNIQUES. THE RESULTS ARE COMPUTED ON A 1000 PAGE DATASET USING A 60:40 SPLIT FOR TRAINING AND TESTING. (*) THE ERROR RATES FOR TESSERACT OCR FOR THESE LANGUAGES ARE VERY POOR TO BE REPORTED.

	Character Error Rate(CER)			Word Error Rate (WER)		
Language	**Our Method**	**Char. OCR [1]**	**Tesseract [17]**	**Our Method**	**Char. OCR [1]**	**Tesseract [17]**
Hindi	1.80	11.34	26.67	5.72	25.38	42.53
Malayalam	1.16	4.16	*	6.08	27.74	*
Kannada	1.01	11.61	*	10.83	51.63	*
Tamil	1.54	16.69	*	13.23	52.56	*
Telugu	1.57	20.63	32.95	9.08	51.49	72.11
Gurumukhi	2.11	6.01	*	7.06	20.64	*
Bengali	2.05	8.06	*	6.80	29.8	*

applicable to newer languages for which OCRs are unavailable. Figure 4 shows some of the qualitative results. We show the words which our method has recognized correctly along with words which our method has failed to recognize. For failure cases, we show the recognized text (in red color) along with the word image. It is to be specially noted that the current work does not use any language models or any other post-processing techniques.

For all these experiments, we have used LSTM size of 50 and number of hidden layers is set at 3. We have conducted our experiments on mid-level desktop PC having 16GB RAM and a 2.3GHz processor. On an average training was conducted for 50 epochs where each epoch took around 1.8Hrs to complete. During testing, recognition of words in a page took 1.57 seconds on average, considering 250 words in a page.

A. Discussion

In case of Indian languages, the average length of the words is quite high which is partly because of the inflectional nature of the language. We verified how our method performs in terms of increase in word lengths. Figure 5 shows such a study where we check the rate at which the mean edit distance error changes with respect to the size of the word. As expected the error is increasing but it does in a slow fashion. In general longer words are rare and the probability of its presence in the training set is low. But since our method uses the context of sub-sequences, the chances are lesser that these rare words also have rare sub-sequences. In contrast a character level OCR does not use any contextual information and would fail miserably for longer words. Figure 2 lists some of the confusing classes present in different languages. Use of contextual information also helps in avoiding mistakes for these cases. Figure 3 shows one of the degraded word image from Tamil language where our method did a correct prediction. As can be seen in the image, isolated character recognition becomes tough due to presence of extreme degradations. However, we use larger contextual information to correctly predict the sequence. The results in Figure 4 shows example images where our method was able to re-arrange the UNICODE sequences correctly.

B. Failure Analysis

The errors shown in Figure 4 are mainly because of missing pixels, *matra's* etc. As one can notice, for Hindi (second error image) there is a strike through in the image which disturbed the profile features resulting in misclassification.

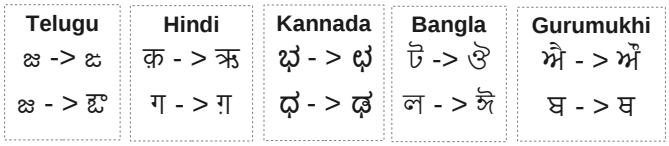

Fig. 2. Confusing Labels: Characters which are frequently confused due to visual similarities.

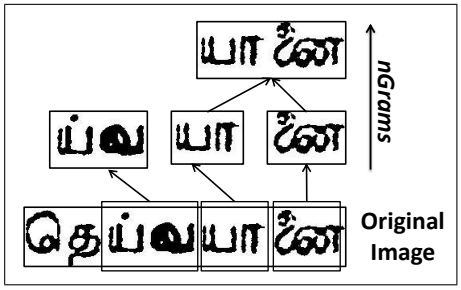

Fig. 3. Degradation Analysis: An example showing successful recognition of our method for a degraded Tamil word image.

In case of Malayalam (first error image), the font resulted in a combination of the last two symbols and produced a valid another symbol as shown in the recognized text. Other common errors were reported in numerals and special symbols such as comma, quotes and fullstops (punctuation). This is because the size of punctuation is much small when compared against the size of entire word. Using better features or looking specifically for punctuation in word image might be able to solve this issue. Degradations in document images are the biggest challenges which will need better features and good amount of post-processing using language models.

V. CONCLUSION

A single framework which can be used to perform robust recognition of multiple Indic scripts have been presented in this paper. This is achieved by considering the problem of OCR as that of transcription of sequence of features to the text sequence. Such an architecture helps us in bypassing the challenges present in UNICODE generation, especially the rearranging rules. By directly predicting UNICODE text, we also avoid the possibility of classifier generating invalid UNICODE sequences (like matras occurring in beginning of a word or a consonant occurring between two vowels etc.). By performing recognition at word level, we also avoid the

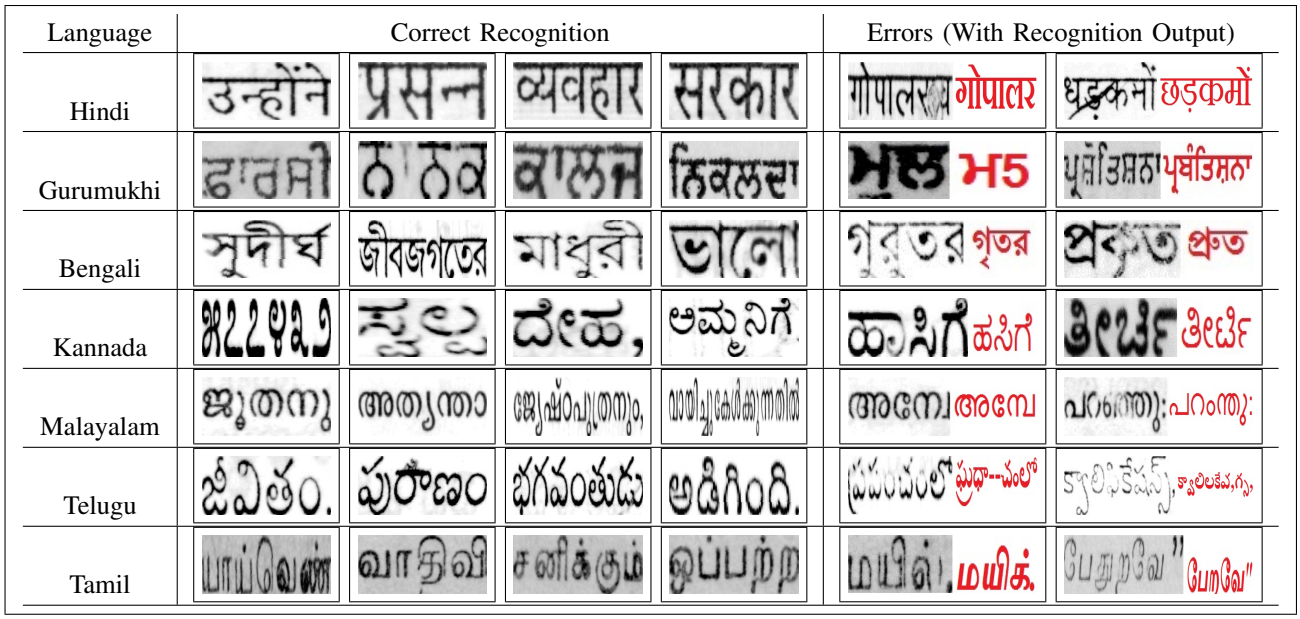

Fig. 4. Qualitative Results: Success and failure examples from different languages are shown in above figure. We show corresponding text recognized by our method along with failed examples.

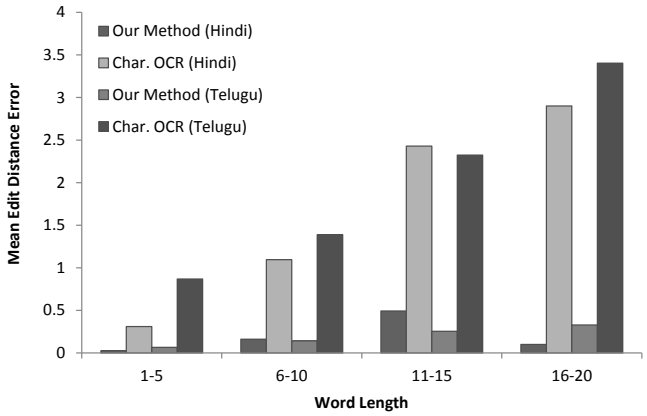

Fig. 5. Mean edit distance v/s word length: Analyzing performance of our method (on Hindi and Telugu) and Char. OCR in terms of mean edit distance error of words recognized w.r.t word length.

challenges due to degradations like cuts/merges etc. Also, the average edit distance for words having large word length is also significantly less. Note that no language models or any other post-processing modules were used to achieve these results. Our OCR is integrated with a web based system. Such a mechanism will also help us in obtaining examples from the users which can be used to further improve the OCR system.

In future, we would extend our system to more Indic languages like Marathi, Urdu etc. We would also like to improve the word recognition rates to facilitate robust search and retrieval over the entire corpus. Such a system should also have efficient post-processing module to further boost the performance.

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*.

[2] C. V. Jawahar and A. Kumar, "Content-level Annotation of Large Collection of Printed Document Images," in *ICDAR*, 2007.

[3] N. Sankaran and C. V. Jawahar, "Recognition of Printed Devanagari Text using BLSTM Neural Network," in *ICPR*, 2012.

[4] 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.

[5] A. Ul-Hasan and T. M. Breuel, "Can we Build Language-Independent OCR using LSTM Networks?" in *J-MOCR Workshop,ICDAR*, 2013.

[6] T. Breuel, A. Ul-Hasan, M. A. Azawi, and F. Shafait., "High-Performance OCR for Printed English and Fraktur using LSTM Networks," in *ICDAR*, 2013.

[7] N. Sankaran, A. Neelappa, and C. V. Jawahar, "Devanagari Text Recognition: A Transcription Based Formulation," in *ICDAR*, 2013.

[8] C. Jawahar, A. Kumar, A. Phaneendra, and K. Jinesh, "Building Data Sets for Indian Language OCR Research," in *Guide to OCR for Indic Scripts*, ser. Advances in Pattern Recognition. Springer London, 2010.

[9] U. Pal and B. Chaudhuri, "Indian Script Character Recognition: A Survey," *Pattern Recognition*, 2004.

[10] R. Jain, V. Frinken, C. V. Jawahar, and R. Manmatha, "BLSTM Neural Network Based Word Retrieval for Hindi Documents," in *ICDAR*, 2011.

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

[12] A. S. Bhaskarabhatla, S. Madhvanath, M. N. S. S. K. P. Kumar, A. Balasubramanian, and C. V. Jawahar, "Representation and Annotation of Online Handwritten Data," in *IWFHR*, 2004.

[13] P. S. Natarajan, E. MacRostie, and M. Decerbo, "The BBN Byblos Hindi OCR system," in *DRR*, 2005.

[14] K. S. S. Kumar, S. Kumar, and C. V. Jawahar, "On Segmentation of Documents in Complex Scripts," in *ICDAR*, 2007.

[15] 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.*

[16] A. Graves, S. Fernández, F. J. Gomez, and J. Schmidhuber, "Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks," in *ICML*, 2006.

[17] Tesseract Optical Character Recognition Engine. [Online]. Available: http://code.google.com/p/tesseract-ocr/