Recognition of Books by Verification and Retraining

Neeba N. V. and C.V. Jawahar
Centre for Visual Information Technology,
International Institute of Information Technology, Hyderabad, India - 500 032

Abstract

The problem of character recognition in a book should be formulated significantly different from that of a single page or word. An ideal approach to design such a recognizer is to adapt the classifier to the font and style of the collection. In this paper, we propose an adaptation framework to recognize characters in a book with a learning framework. In the proposed system, the post processor verifies the output of the recognition module, which is further used for learning and thus to improve the performance over iteration. Experiments are conducted on about 500,000 annotated symbols from five books in Malayalam (an Indian language). We achieve an average improvement of 14% *in classification accuracy.*

1. Character Recognition

Optical character recognition (OCR) is a well researched and mature area in the literature [1, 2]. Most, if not all, of these studies focused on recognition of isolated pages or words. With the emergence of digital libraries, in recent years, recognition of a complete book (or a document image collection) has become important [3, 4, 5]. For machine understanding and language processing of digitized document images, it is important to convert them into a textual form using an OCR. We argue that the recognition of a book, as a large collection of symbols, is considerably different from that of a word or a page. This is due to the fact that books provide rich additional information that could be used for improving the recognition rates of character classifiers and OCRs.

The major component in a typical OCR system is a pattern classifier, with very high performance to recognize isolated symbols. Any error at this stage can get propagated, if not avalanched, into the next phase. A classifier is often trained offline using labeled samples in multiple fonts and styles. With unseen fonts/styles, the performance may deteriorate. When it comes to the recognition of a large collection, such as a book, traditional OCRs may repeat similar mistakes across pages. Or else an OCR designed for isolated pages need not *learn* to improve the performance over time. The advantage of books from a recognition point of view is that, books are often typeset in a single font and style. This implies that, we can use the first few pages to obtain better performance over the rest of the collection.

We enhance the recognition/classification rates of our Book-OCR by *verification* and *retraining*. New training data is introduced into the system without any manual intervention. This is done with the help of a dynamic programming based verification module. We employ an automatic learning framework to get more training samples with the help of a verification module. Thereafter, we employ these samples to improve the performance of the classifier by retraining. Our primary contribution is a novel procedure that is specially suited for the recognition of books. We obtain an average performance improvement of around 14% in classification accuracies. We obtain performance enhancement at the cost of additional computations during the verification and retraining.

The experiments are conducted on samples from five annotated books in Malayalam script(an Indian script). At present there is no robust commercial OCR system available for this language. Languages like English employ a dictionary to correct errors in the recognition phase. However, dictionary based post-processing techniques are not feasible for highly inflectional languages like Malayalam.

A verification scheme based on a dictionary based approach was employed in [4] for the adaptation of character recognition. Recently, another step towards the whole book recognition is reported in [5], using mutual entropy based model adaptation, which uses linguistic constraints to calculate the posterior probabilities of word classes. In our implementation, we replace the conventional post-processor with a verification scheme.

2. Overview of the Book Recognizer

We present a novel and practical approach, for the recognition of symbols from a large collection of documents. Our method employs a data-driven adaptation method to enhance the performance, specific to a particular collection. Training the system with the samples from the same collection to be recognized, is an obvious method to improve the performance of the system. However, collecting the samples from the same environment and labeling them, each time when a new book needs to be recognized, is impractical. We propose an automatic procedure for getting labeled samples and allowing the system to adapt to the same. This is achieved with the help of a high performance verification module, which acts as a post-processor in the system.

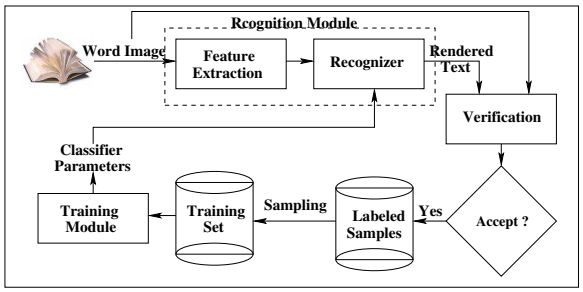

Figure 1. Overview of the proposed book recognition scheme.

The input to the recognizer is a word image. We assume that preprocessing and segmentation of document images upto word level is available. The overall working of the system is as follows. The base recognizer parses the connected components in the word image and identifies the sequence of symbols to be recognized. The recognizer classifies each symbol and returns a class-id associated with it. The class labels are converted back to a Unicode representation. Note that for the Indian scripts, the basic symbols we employ are different from the Unicode[6]. Verification module validates the recognized word *visually*. (Note that a typical post processor validates using a language model.) At this stage, a new set of automatically labeled samples are generated (more details in the next section). In the next iteration, a subset of the labeled samples(depending on the sampling rate chosen) are added to the training set and the classifier is re-trained with the modified set. We repeat the process until, the required accuracy is achieved or the performance improvement in an iteration is less than a threshold. The procedure is summarized in Algorithm 1. Retraining the system in this manner, creates a new improved classifier. We use this new multi-class classifier for further recognitions. This process is repeated until the accuracy of the classifier saturates. The overview of the recognition process is shown in Figure 1.

Algorithm 1 Algorithm for adaptation in book recognizer.

1: Input: Word images from the books.
2: Find the connected components(symbols).
3: Recognize each symbol using the classifier and get the class-id.
4: Map the class-ids to Unicode(text) and render the text to obtain a word image.
5: Verification module takes the input word image and the rendered word image to matches both with a dynamic programming based algorithm.
6: The matched samples from the original word image are stored in a labeled images database.
7: The samples are randomly selected from the database (according to the sampling rate selected) and added to training set.
8: Train the classifier with new samples and use this new classifier for further classification.
9: Repeat the steps 2-7 until the performance improvement is less than a threshold.

With the availability of huge computational power, machine learning techniques are finding new applications in document image analysis [7]. In this work, we give more importance to the effectiveness in terms of accuracy of the system than the time consumed for retraining and recognition. At this stage, it is possible to assume that the machines used for recognizing books have enough computational power to do some additional computations so that the performance of the recognizer can improve.

Sampling is an important aspect of building the training set. The sampling scheme employed in the system randomly selects the samples from the labeled samples database, and add to the training set. The sampling rate can be specified by the user. While generating more training data, we give priority to the samples which got misclassified initially and correctly classified in subsequent iterations. This is done by restricting the sampling from newly labeled samples in the labeled samples database at each iteration.

3. Verification Scheme

A dynamic programming (DP) based verification module acts as the post-processor for our system. In contrast to the conventional post-processing methods based on the dictionary look-up, our method uses an image based matching technique for verification. The input of the verification module is a word image and the corresponding Unicodes from the recognizer. The output Unicodes are rendered to get an image, which

is matched with the input word image. The output of the verification module is a set of labeled samples from the word image. Note that the purpose of the verification module is to provide labeled data for retraining the classifier in the next iteration and thus improving the performance of the recognition module. We need to make sure that correct samples are selected in the training set.

In the DP table, each cell is filled with the matching cost of connected components in the rendered and original word image. In our DP table, diagonal elements represent the *one-to-one* matching of the symbols from the original image and corresponding rendered image. Consider a simple case, where the input word image does not have any cut or merge. In this case a high score in the diagonal element refers to a mismatch and a low score refers to the match, as shown in Figure 2. A match in a diagonal cell refers to the correct classification in the recognition and the mismatch refers to the misclassification. Though the focus of this work is not to

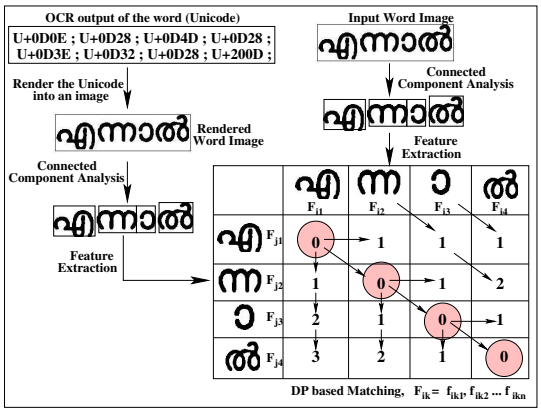

Figure 2. An Example of a dynamic programming based verification procedure. Word image is matched with an image rendered out of the recognized text.

recognize the degraded characters with cuts and breaks, our verification algorithm identifies such samples, and avoid them in the sampling process. A cut or merge results in the non-diagonal path in the dynamic programming. In the case of cuts, two or more image components may correspond to one rendered component. The DP that we employed is different from the popular DP algorithm (used for string matching) in the following aspects: (a) We obtain cuts and merges instead of deletions and insertions in a normal DP algorithm. (b) The matching is done in the image domain to find the match scores in DP. (c) The computation of matching cost is different for non-diagonal elements. Algorithm 2 gives the details of the DP based word matching process.

The verification module uses simple and structural features. This module performs well, even if the characters are similar. In our implementation, we assume that a character can get cut only into two pieces and the merges can happen only between two characters.

Algorithm 2 Algorithm for Verification as post-processor.

1: Input: Word image and the corresponding text from the Recognizer.
2: Render the text to get a word image.
3: Find the connected components of both original and rendered word images.
4: Create the Dynamic Programming based Cost table, by matching the symbols.
5: Fill the cost values in the table $C(i, j)$ as,

$$C(i,j) = \min \begin{cases} C(i-1, j-1) + MC(S_i, K_j) \\ C(i-1, j) + MC((S_{i-1}, S_i), K_j) \\ C(i, j-1) + MC(S_i, (K_{j-1}, K_j)) \end{cases}$$

where, $MC(S_i, K_j)$ is the matching Cost of symbol S_i in the text(rendered as image) with symbol K_j in the original image.
Matching Cost will be 0 if the two symbols matches, otherwise matching cost will be 1.
6: Get the matching String by reconstructing the path, by following the minimum cost path.
7: Output the text that caused the minimum path as recognized text.

4. Results and Discussions

We have conducted our experiments on five Malayalam books. For the automatic evaluation of performance, all these books are annotated *apriori* [8]. The quality of the images in the corpus vary widely. These books contain more than $500,000$ symbols to recognize.

Book title	# Pages	# Words	# Symbols
Book 1	96	11,404	74, 774
Book 2	119	20,298	147,652
Book 3	84	10,585	83,914
Book 4	175	21,292	152,204
Book 5	94	12,111	92,538

Table 1. Details of the books used for the experiments.

In this work, our focus is on improving the classification accuracy of these symbols rather than recognizing degraded images. Malayalam has 56 basic alphabets in its character set, in addition to a large number of conjunct characters. We have considered a total of 205 classes for our experiments, which includes, all the

basic alphabets, popular conjunct characters, punctuations, numerals etc. Table 1 summarizes the details of the books used for the experiments.

Input to our system is a set of binary document images. After word-level segmentation, the connected components are identified in each of the word images. They form the basic symbols for classification. The symbols are then scaled to 20×20 pixels size keeping the aspect ratio unchanged. We then use PCA for the feature extraction. The scaled image is converted into a $1D$ array of 400 pixel values which is mapped to the PCA feature space of length 350. Pair-wise SVM classifiers with linear kernel is employed as the basic classifier. The initial classifier is trained with five popular fonts. The books used for the experimentation are typeset in fonts somewhat different from that used for training.

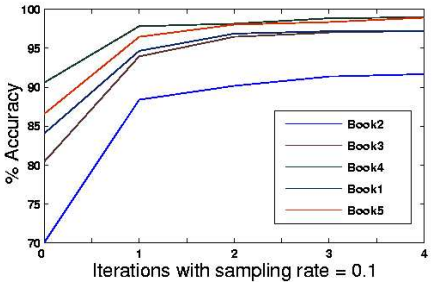

Figure 3. Improvement in the performance of a book, with sample rate = 0.1.

Our experimental results on different books are shown in the Figure 3. The performance improvement obtained in the books depends on the quality of the book. This is why the maximum accuracies obtained vary across the books. With sampling rate 0.1 in one case (Book 3:*Thiruttu*), we obtain a performance improvement of 21.66% (from 70.02 to 91.68) and in another case (Book 4:*ValmikiRamayanam*) we obtained 8.43% (from 90.58 to 99.01). The average percentage improvement is 13.61. In each iteration we recognize the complete book.

It is observed that, in the first iteration the performance improvement of the classifier is significantly high. As the iteration progresses the rate of improvement decreases. Even though the SVM classifier is considered as a strong and stable classifier, addition of these training samples practically results in significant change in the decision boundaries. The sampling rate also affects the learning rate of the system. We conducted the experiments by varying sampling rate. It was felt initially that if the sampling rate is high the OCR will learn fast. However, in our experiments the change in learning rate with the change in the sampling rate was marginal (Refer Table 2). This is because additional samples of the same font/style do not improve the classifier significantly. This also means that, we can select a low sampling rate and reduce the training time.

# Iteration	0.01	0.05	0.1	0.3
0	80.42	80.42	80.42	80.42
1	93.28	94.33	93.96	93.94
2	96.46	96.80	96.47	95.35
3	97.41	97.59	97.01	96.28
4	97.52	97.69	97.26	96.46

Table 2. % Accuracies obtained with varying sampling rate for the Book 3.

5. Conclusion

In this paper, we propose a novel system for adapting a classifier for recognizing symbols in a book. We employed a verification module as a post-processor for the classifier, and make use of an automatic learning framework for the continuous improvement of classification accuracy. We obtain an average improvement of 14% in classification accuracy. This demonstrates that the proposed approach is promising for the recognition of books.

References

[1] Y. Xu and G. Nagy, "Prototype extraction and adaptive ocr," *IEEE Trans. on PAMI*, vol. 21, no. 12, pp. 1280–1296, 1999.

[2] C. L. Liu and H. Fujisawa, "Classification and learning methods for character recognition :Advances and remaining problems," in *Machine Learning in Document Analysis and Recognition*, pp. 139–161, 2008.

[3] K. P. Sankar, V. Ambati, L. Pratha, and C. V. Jawahar, "Digitizing a million books: Challenges for document analysis," in *DAS*, pp. 425–436, 2006.

[4] M. Meshesha and C. V. Jawahar, "Self adaptable recognizer for document image collections," in *PReMI*, pp. 560–567, 2007.

[5] P. Xiu and H. S. Baird, "Whole-book recognition using mutual-entropy-driven model adaptation," in *Document Recognition and Retrieval XV. Proc. of SPIE*, 2008.

[6] U. Pal and B. B. Chaudhuri, "Indian script character recognition: a survey," *PR*, vol. 37, no. 9, 2004.

[7] S. Marinai and H. Fujisawa, eds., *Machine Learning in Document Analysis and Recognition*, vol. 90 of *Studies in Computational Intelligence*. Springer, 2008.

[8] C. V. Jawahar and A. Kumar, "Content-level annotation of large collection of printed document images," in *ICDAR*, pp. 799–803, 2007.