Self Adaptable Recognizer for Document Image Collections

Million Meshesha and C.V. Jawahar

Center for Visual Information Technology,
International Institute of Information Technology,
Hyderabad - 500 032, India
jawahar@iiit.ac.in

Abstract. This paper presents an architecture that enables the recognizer to learn incrementally and, thereby adapt to document image collections for performance improvement. We argue that the recognition scheme for a book could be considerably different from that designed for isolated pages. We employ learning procedures to capture the relevant information available online, and feed it back to update the knowledge of the system. Experimental results show the effectiveness of our design for improving the performance on-the-fly.

1 Adaptable OCR System

The success of document image indexing and retrieval in the newly emerging digital libraries considerably depends on the availability of robust OCRs that can take care of the diversity in the document image collections. Performance of the state of the art OCRs are not very encouraging for these collections [1,2]. Recent study by Lin [3] shows that document recognition research is still in great need for better accuracy and reliability, as well as for effective information retrieval and delivery. We need a recognition system that is capable of intelligently adapting to the characteristics of documents of interest, and improving the performance over time. Machine learning offers one of the most cost effective and practical approaches to the design of pattern classifiers for a broad range of pattern recognition applications like character recognition [4]. Learning algorithm could be supervised, unsupervised or reinforcement-based [5]. Supervised techniques have been successfully demonstrated for character recognition application as offline training in OCR systems. However applicability of semi-supervised and reinforcement learning algorithms are not yet explored in their full potential.

Most of the recent research in OCR has been centered around building fully automatic, high performing intelligent classification systems with good generalization capability [6]. Intelligent OCRs with excellent performance on a given page are reported for Latin scripts [7]. However, when it comes to the suitability of converting an old book to text and providing a text-like access, even the present day OCRs are found to be insufficient [1]. The performance of these recognizers decline as the diversity in the collection of documents (with unseen

A. Ghosh, R.K. De, and S.K. Pal (Eds.): PReMI 2007, LNCS 4815, pp. 560–567, 2007.

fonts and poor in quality) increases. We argue that an adaptive recognition system, which can learn from its experience and improve its performance over time, is better suited for such document collections (that vary in printing and quality).

In this paper, we argue that:

1. The technique for recognizing a book (a reasonably large collection of documents in single font and consistent formatting) can be different from OCRs that are designed to be part of scanner drivers or meant for isolated pages.
2. The notion of generalization from a small set of training samples, which is critical to the design of pattern classifiers, need not be the only performance goal. One can positively look for 'overfitting' to a book, as long as the recognition engine can provide better results on that specific collection. In general, the multimodality of the data distribution need not be completely modeled; but can be accepted as a reality.

An approach for designing a semi-automatic adaptive OCR for document images in digital libraries is reported in [8]. The OCR is designed to support an interactive retraining (based on users feedback) for performance improvement. In this paper, we present an intelligent and self adaptable OCR that employs machine learning algorithms for validation, labeling, sampling and incremental learning in a recognition cycle. Post-processor based feedback mechanism is enabled to provide additional knowledge to the system.

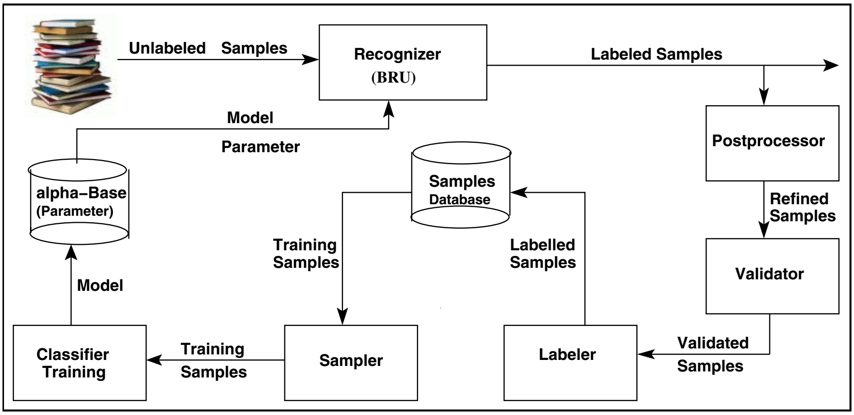


Fig. 1. An architecture of OCR learning framework employed for the recognition of document collection

2 Learning Schemes

There are possibly two ways in which the continuous performance improvements can take place in recognizers for document collections in digital libraries.

1. Learn from the direct or indirect user feedback (as in the case of relevance feedback models, search engines etc.) captured during the search.
2. Improve the performance of recognizers by adapting to a specific collection by absorbing the input from language models (e.g.. Dictionaries).

In this paper, we do not model the user interactions, and follow the second approach, where knowledge from the post-processor is used for developing new labeled examples, and thereby improving the recognition accuracy. In the conventional OCRs also, language models are integrated for improving the performance of the base-classifier. However they are not designed to learn from their experience. If a word which is misclassified once is given to the base classifier, it repeats the mistake again. However, we are interested in a framework, which will make the classifier intelligent and correct the mistake in the future.

Algorithm 1. Recognition of Text in Document Collections

1: Document images are preprocessed and words are extracted for recognition.
2: BRU outputs the recognized text.
3: Post-processor checks the validity of the word based on some language model and outputs the corrected word, if found to be invalid.
4: Validator visually validates the result of post-processing. By matching the rendered text and word image, visual similarity is carefully computed. Errors introduced by the post-processor results in outliers.
5: Those with visually similar appearance are considered as new training samples. Semi-supervised learning is used in labeling the new samples.
6: Bagging is used to create a sample set and classifier is trained (preferably incrementally) to obtain the new set of parameters and stored in α-base.
7: Base-classifier is incrementally learn for knowledge updation.

Overview: Architecture of the proposed recognition system is shown in Figure 1, and also explained in Algorithm 1. Given a word image for recognition, the Basic Recognizer Unit (BRU) converts it into text. A post-processor is then used to rectify/verify the recognized word. In our present implementation, we use a simple dictionary-based post-processor which is designed following a reverse dictionary approach with the help of a trie data structure. Our Basic Recognition Unit is an SVM-based classifier trained offline on few *apriori* available synthetic training examples (that are prepared in single font, style and size) by extracting vector representation of the entire image. In this work, the conventional open-loop system of classifier followed by post-processor is closed by automatically generating training data from the “test” image and retraining (in fact incrementally modifying) the BRU. Before passing mis-recognized samples as new training datasets we run a validator to detect outliers that deviate from the original sample. Samples that are similar to the original ones are labeled using labeler and transferred to the sample database. This system, thus, has a framework for the creation, validation, labeling and use of mis-recognized samples as new training examples for performance improvement over time.

This character recognition system is highly data-driven. It employs feedback for performance improvement. Initial recognition errors are immediately detected and corrected. This technique is also useful when sufficient training samples or *apriori* knowledge is lacking (for example recognizing in a new font/style). The implementation of this approach could be computationally intensive. However, today's OCR environment especially the ones for digitizing books and large document image collections can afford to do so.

Automatic Preparation of Labeled Data: One of the basic problem of the learning process in a dynamic environment is getting high quality new training samples. Once the BRU recognises a word and outputs a textual representation, post-processor verifies the validity of the word with the help of a dictionary (or a language model) and either accepts the word as valid or corrects with an alternative. There are six possible situations.

1. BRU correctly recognizes a word and post-processor accepts it as valid word.
2. BRU correctly recognizes, but post-processor fails to accept it as valid and suggests an alternative.
3. BRU makes a mistake, and post-processor corrects to the right word.
4. BRU makes a mistake, and post-processor corrects/modifies to a wrong word.
5. BRU correctly recognizes, but post-processor modifies to a wrong word.
6. Post-processor fails to suggest an acceptable alternative to a text provided by BRU.

We employ the knowledge from the post-processor in creating a new set of labeled examples. For this, we validate the result of the post-processing, by matching in the image space. *This is more of verification rather than recognition.* Given the text, it is rendered into an image, and matched in the image space. Let $p_1, p_2, \ldots p_m$ and $q_1, q_2, \ldots q_n$ be a set of feature vectors, extracted by scanning (column-wise) the vertical strips of the rendered text and word images, where m and n are width of a given words. They are aligned using dynamic time warping (DTW) for similarity measure. If all the symbols match (one-to-one), then we have achieved the labeling of all the connected components in the document image (situation 1). When only some of the symbols match, they are labeled and the rest of the symbols are treated as unlabeled. In short, at the end of the first phase of validation using a DTW-based algorithm, most of the components from a given document image gets labeled automatically and gets added to the database of training examples. The rest of the samples are treated as unlabeled. Note that, at this stage, our interest is limited to improving the performance of BRU, rather than addressing the problem of merges and splits in the document images.

For each unlabeled data x_i we attach probabilities p_{ij} of belonging to class i, i.e. $p_{ij} = prob(x_i \in w_j)$. This is an initialization. These probabilities are then iteratively improved by an Expectation Maximization (EM) [9] based formulation. The E-Step uses the current parameter estimates to find the best probabilistic labels for class membership using a multivariate normal (Gaussian) distribution. The M-Step then refines the parameters to maximize the total likelihood. The

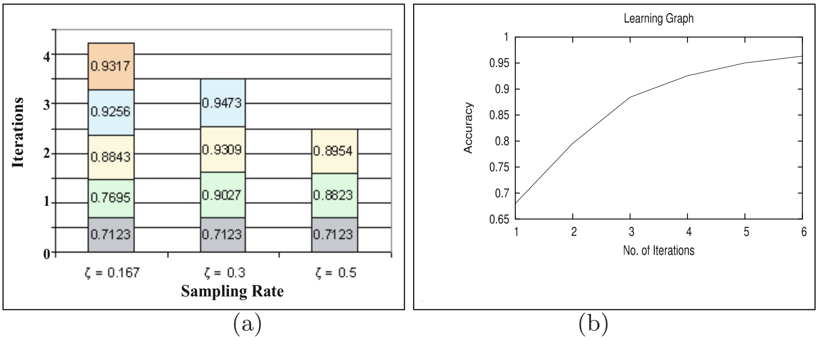


Fig. 2. (a) A change in accuracy and learning rate at different sampling rate (ζ). (b) An improvement in the performance of the recognizer through learning from poor quality documents.

steps are iterated until the change in parameter values falls below some predefined threshold. In a way, it allows to put the unlabeled examples into a cluster with most of the samples already labeled. When most of the samples are labeled, it converges in one step, and assigns label to the unlabeled samples.

Sampling: The ability to actively select the most useful training samples is an important aspect of building an efficient classifier. Boosting and bagging are being increasingly used for this purpose [5,10]. Boosting uses all instances of the datasets at each iteration, but associates a weight for each sample. Bagging, on the other hand, takes the available training samples and generates a new sample set by selecting them randomly and with replacement. In our implementation, training examples are generated throughout the recognition process. We need to use these samples as new training datasets in subsequent iterations; as a result of which we use bagging for sampling. For each session $k = 1, 2, \ldots, n$, new training set of size N is sampled from the database. Bagging works as follows. Consider a training dataset $D = d_1^{c_i}, d_2^{c_i}, \ldots, d_m^{c_i}$ where m is the total samples available in class c_i. Bagging selects a subset of representative samples randomly from the available training sample collections that are accumulated through feedback. In each session k, a re-sampled training set D^k is built for constructing/training a classifier C^k. Classifier C^k has better accumulated knowledge than the previous C^{k-1} increasing its applicability to the given document collection.

Incremental Updation: We use SVM classifier [5] as the basic recognition unit, and employ an incremental learning approach [11] to train it. By identifying the decision boundary with maximal margin, SVM results in better generalization during classification. Margin maximization leads to an optimization problem the solution of which is expressed uniquely in terms of support vectors s_i. An input pattern x is classified into class $y \in \{-1, +1\}$ according to the decision function:

$$y = sgn(\sum_i \alpha_i y_i K(s_i, x)) \tag{1}$$

Table 1. Performance improvement with the use of incremental learning vs. retraining during the learning process.

Iterations	Retraining	Incremental
1	0.652475	0.652475
2	0.867845	0.882446
3	0.898620	0.907383
4	0.901226	0.910813
5	0.929970	0.939294
6	0.941537	0.948294
7	0.943962	0.959026
8	0.952096	0.959026
9	0.952096	0.959026
10	0.952096	0.959026

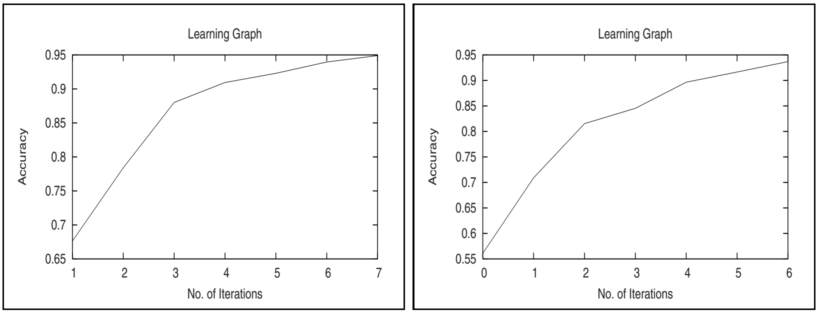
which takes the form of a linear combination of kernels $K(s_i, x)$ weighted by training labels y_i and coefficients α_i. The coefficients α_i are nonzero only for training data that are support vectors, so that $\sum_i \alpha_i y_i K(s_i, x)$ is sparse and the support vectors capture the relevant information present in the training data.

Incremental SVM works as follows. The representation of the data seen so far for each class is given by the set of support vectors describing the learned decision boundary. These support vectors are combined with the new incoming datasets to provide the training data for the incremental step. The incremental step then updates the solution for addition of a single training sample x_i by incrementing the coefficient i and simultaneously adjusting previously assigned coefficients $\alpha_j(j < i)$ on the present and all previous training samples.

3 Implementation, Results and Discussions

Our implementation in c/c++ is tested on document image collections. The design of our system is based on a multi-core approach [2]. Keeping the futuristic large scale computational applications, it is implemented as layers with plugin interfaces for modules to ease replacing one module with another. SVM based classifier could be modified as a cascaded classifier combination with out major changes in other parts of the code. Each module internally can decide on multiple algorithm implementations of the same functionality that may be interchanged at run-time. This helps in selection and use of an appropriate algorithm or a set of parameters for a document collection or script. The system allows transparent run-time addition and selection of modules thereby enabling the decoupling of the application and the plug-ins.

The learning scenario involves post-processor based feedback mechanism to update the knowledge of the recognizer. Machine learning procedures are integrated for labeling, sampling and validating the new training samples collected online. Scanned images are segmented into words and submitted for recognition in a batch. Recognition results are post-processed to resolve ambiguity among



(a) (b)

Fig. 3. Recognition accuracy of the OCR through learning in presence of (a) font variations and (b) style variations

the candidate words. Mis-recognized words are validated and fedback as new training datasets for learning incrementally. The labeler is used to split these words into components and assign their membership category (class). The sampler selects subset of these datasets and pass them for learning. Sampling speeds up the learning rate η if proper sampling rate ζ is used. As shown in Figure 2(a) the learning rate and improvements in recognition accuracy are related with the sampling rate used. Initially the accuracy jumps from 71.23% to 76.95%, 90.27% and 88.23% for sampling rate 0.167, 0.3. 0.5 respectivelly. We can observe that the number of iterations decrease as the sampling rate increase, but with better accuracy obtained at sampling rate ζ of 0.3. This shows that sampling at 30% of the original training datasets speeds the learning rate with better improvements in recognition results. This is only an empirical observation. We use this sampling rate for our experimentation.

Once new training samples are selected, incremental SVM classifier is used for updating the knowledge of the recognizer. Table 1 presents performance of incremental learning vis-a-vis retraining. The result shows that the two approaches have comparable performance. However, the use of incremental learning has further advantage in terms of both computational time and space complexity. In a way, it eases the implementation of SVM classifier on large datasets.

We validate the performance of the learning framework on real-life printed document images (of English) that are (i) poor in quality and (ii) varies in fonts and styles as depicted in Figures 2(b) and 3, respectively. The result demonstrate the advantage of our learning strategy for performance improvement. Because of the quality and printing variations in document images, a very low recognition result is obtained at the initial stage, which amounts to less than 70%. Within few iterations of learning, however, the recognition accuracy improved, on the average, to 95%. Most of the recognition errors happen since characters do not always look the way they should because of degradation or printing variations. A character's hole is filled, e.g. 'o', 'u' and 'n' are recognized as 'dot', some part of the character (e.g. dot of 'i') is missing and recognized as '1' or 'l', etc. Different fonts produce a character with different shape that may visually similar with other character. It has been observed that the recognizer is able to adopt to the given document images based on the additional samples of confusing components that are fedback for learning. In this way, the learning framework can enable the OCR to learn from its experience and adapt to varying document image collections in printing and quality. Further work is needed for (i) Extending this framework for many of the complex Indian scripts (ii) Addressing the segmentation errors at various stages. The present design assumes that segmentation (of pages into words as well as words into recognizable symbols) is available. This assumption needs to be relaxed in future.

4 Conclusion

We have presented a novel approach for learning during the recognition of document image collections. The architecture integrates advanced learning procedures that automatically interacts and pass feedback for further learning. This enables the OCR to easily accumulate knowledge for performance improvement. This strategy is promising for the recognition of large digitized document images in applications, like digital libraries. Experiments are ongoing to validate our approach on diverse collections with changes in script, font, etc.

References

1. Feng, S., Manmatha, R.: A hierarchical, HMM-based automatic evaluation of OCR accuracy for a digital library of books. In: Joint Conference on Digital Libraries (JCDL), pp. 109–118 (2006)
2. Sankar, P., et al.: Digitizing a million books: Challenges for document analysis. In: Proc. of the Seventh IAPR Workshop on Document Analysis Systems, pp. 425–436 (2006)
3. Lin, X.: DRR research beyond COTS OCR software: A survey. In: SPIE Conference on Document Recognition and Retrieval XII, San Jose, CA, pp. 16–20 (2005)
4. Xu, Y., Nagy, G.: Prototype extraction and adaptive OCR. IEEE Transactions on Pattern Analysis and Machine Intelligence 21, 1280–1296 (1999)
5. Hastie, T., Tibshirani, R., Friedman, J.: The elements of statistical learning. Springer, Heidelberg (2001)
6. Nagy, G.: Twenty years of document image analysis in PAMI. IEEE Transactions on Pattern Analysis and Machine Intelligence 22, 38–62 (2000)
7. Kahan, S., Pavlidis, T., Baird, H.S.: On the recognition of printed characters of any font and size. IEEE Transactions on Pattern Analysis and Machine Intelligence 9, 274–288 (1987)
8. Rawat, S., et al.: A semi-automatic adaptive OCR for digital libraries. In: Proc. of the Seventh IAPR Workshop on Document Analysis Systems, pp. 13–24 (2006)
9. Ivanov, Y., Blumberg, B., Pentland, A.: Expectation maximization for weakly labeled data. In: Proc. of the Int. Conf. on Machine Learning, pp. 218–225 (2001)
10. Iyengar, V.S., Apte, C., Zhang, T.: Active learning using adaptive resampling. In: Sixth Int. Conference on Knowledge Discovery and Data Mining, pp. 92–98 (2000)
11. Diehl, C., Cauwenberghs, G.: SVM incremental learning, adaptation and optimization. In: Proc. IEEE Int. Joint Conf. Neural Networks, pp. 2685–2690 (2003)