2009 10th International Conference on Document Analysis and Recognition

Classifying foreground pixels in document images

Prateek Sarkar, Eric Saund, Jing Lin
Perceptual Document Analysis
Palo Alto Research Center, Palo Alto, CA, USA
{psarkar,saund,jlin}@parc.com

Abstract

We present a system that classifies pixels in a document image according to marking type such as machine print, handwriting, and noise. A segmenter module first splits an input image into fragments, sometimes breaking connected components. Each fragment is then classified by an automatically trained multi-stage classifier that is fast and considers features of the fragment, as well as its neighborhood. Features relevant for discrimination are picked out automatically from among hundreds of measurements. Our system is trainable from example images in which each foreground pixel has a "ground-truth" label. The main distinction of our system is the level of accuracy achieved in classifying fragments at sub-connected component level, rather than larger aggregate groups such as words or text-lines. We have trained this system to detect handwriting, machine print text, machine print graphics, and noise.

1 Introduction

Markings in document images may come from a variety of sources. Machine print (text, line graphics, block graphics), hand writing (text, line graphics, shading), stamp marks, shadows (page folds, page curl, punch holes, paper texture) are some of the different sources of marks on a page. Once a page has been scanned, these markings are *flattened* to an image of pixels. To human perception the various kinds of markings still stand out as separate entities. This helps us read documents even in the presence of overlap and clutter. We wish to make the facility of telling different kinds of markings available to automated processes.

Our work is motivated by various usage scenarios. In legal document discovery a well known problem is to quickly identify, from millions of pages, those which have been marked on by hand. In automated data extraction, absence of handwritten marks in a signature box can be flagged as the absence of signature. Telling noise and handwriting apart from machine print can lead to better segmentation for OCR [14].

Various rule-based and image processing techniques have been applied to detection and/or removal of granular noise, line graphics, shadows or machine print text [9, 2]. In commercial packages extraction of machine printed and handwritten text has been heavily engineered for known contexts of document type (checks, forms, letters), language, script, and text-size.

There has been interest, in recent literature, to address the problem through machine learning. The motivation is to improve performance and retargetability. Zheng et al. [14] classify regions of pixels (roughly text words) into machine print text, handwritten text or noise. They construct a Markov Random Field to incorporate neighborhood context, whereas Shetty et al. [13] employ Conditional Random Fields. An et al. [1] aim to classify every pixel. Koyama et al. [8] use local texture features to classify small patches of an image into machine-printed or hand-written. Chen et al. [4] focus on the selecting the right granularity of segmentation a posteriori through a multi-scale hierarchical segmentation and classification scheme.

Classification performance depends on the target granularity. Clearly, it is futile to classify a single pixel without surrounding context, especially in binary images. Although larger entities (words, lines, regions, pages) carry more information and are easier to classify [14], we choose to classify fragments at the connected component or sub-connected component level. Our motivation is twofold. First we want to be able to call out touching markings of different types – so we want to split connected components. The second is to build a useful basic building block (fragment classifier) with the understanding that coarser level decisions (at the level of words, regions, or pages) can be made with much higher accuracy by aggregating the output of our basic tool. To this end we present a new classifier architecture that is fast and accommodates neighborhood context.

2 Problem setup

We make the following broad assumptions to enable easy creation and representation of ground-truth, and a consistent evaluation metric. In this we are heavily influenced by re-

978-0-7695-3725-2/09 $25.00 © 2009 IEEE
DOI 10.1109/ICDAR.2009.252

cent trends [12, 1].

Pixel labels: Each pixel has a single marking category label. This assumption is purely pragmatic. It allows us to represent the ground-truth of an image as another image with an integer label for each pixel. Thus we can store ground-truth and processed output using regular image formats, and use image viewers, loaders, and editors to visualize and manipulate them.

Background pixels: We assign marking categories only to foreground pixels. Currently we assume that black (ink) pixels are foreground, and white pixels are the background (paper). Clearly, both background and foreground pixels need to be analyzed to tell different markings from each other.

Ambiguous pixels: Pixel label ambiguity, when multiple markings overlap, is resolved by *label priority*. In our implementation, "Handwriting" is the category with highest priority. When handwriting overlaps machine print, pixels in the intersection are labeled as "Handwriting". Human labelers still have to make a subjective call to identify pixels in the intersection. Noise labels have the lowest priority.

Evaluation: When comparing two ground-truth label files, or an automatic classification output to ground-truth, we compare the labels at each pixel location and count one error if the two labels differ. Comparing pixel-labels naturally leads to the confusion matrix, and other derivative metrics such as category-wise precision and recall.

Our target marking categories are, in order of disambiguation priority:

• *Handwriting*: Includes text (cursive, print), and graphics.

• *MachinePrintText*: Black-on-white text that is machine printed. Shaded text, or black background for white text is considered *MachinePrintGraphic*.

• *MachinePrintGraphic*: Underlines, arrows, background rules, line-art, bullets, logos, photos, etc.

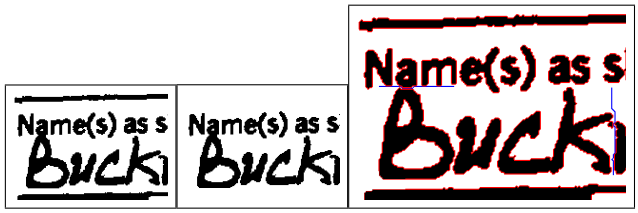
• *ScannerNoiseSaltPepper*: Small granular noise usually due to paper texture, or faulty binarization.

• *ScannerNoiseDarkRegion*: Black pixels due to shadows and darkening such as from paper curl, folds, and holes.

Our goal is to develop an algorithm that will label pixels according to the above marking types, while minimizing the number of pixel-label disagreements with human labelers.

3 Solution Architecture

Our architecture for detecting different kinds of markings consists of a segmenter and a classifier. A *segmenter* partitions the set of foreground pixels in a document image into a list of fragments. A *classifier* then takes each fragment and assigns a category label to that fragment. Our classifier actually returns scores corresponding to different categories of markings, and we simply pick the category with the best score. A downstream application may further interpret the scores in order to make decisions. For example, scores may be aggregated to classify larger regions, or fragments that have handwriting scores above a preset threshold may be highlighted or marked for annotation processing.



(a) (b) (c)

Figure 1. Illustration of our segmentation algorithm.

3.1 Segmenter

The segmenter splits an image into *fragments* – groups of foreground pixels that, we assume, have the same label. Oversegmenting (splitting connected fragments that are one marking type) produces smaller fragments and can make classification more difficult. But undersegmentation produces mixed marking-type fragments and is bound to cause pixel-classification errors. For the trade off, we aim to oversegment rather than undersegment. We rely on connected component analysis, but decide to split some connected components. This is to address overlapping content such as handwriting and machine print graphics in forms.

Our segmentation algorithm consists of two stages. In the first stage horizontal and vertical lines are detected and removed by morphological operations. In the second stage, connected components are found, and some of them are recursively split till they pass a size test. We illustrate this with the example image in Figure 1(a). (b) shows the image with lines removed. Fragments generated by recursive splits are shown in (c) with their boundaries highlighted. The recursion here is just one level deep. The *split-paths* are shown in blue.

The splitting algorithm is based on dynamic programming, and finds the lowest cost split-path that traverses the sub-image to be split. The cost of a split-path is the sum of pixel costs along the path. The pixel costs are positive for foreground pixels (to penalize cutting through foreground), and negative along edges (to encourage edge following). The basic split-path algorithm is very similar to Breuel's [3].

3.2 Fragment features

Each fragment is characterized by a number of features (measurements) that are computed on the fragment and surrounding context. A classifier is trained to classify each fragment into one of the categories of markings, on the basis of these features. We list a few of the features we have implemented. Many other informative features have been

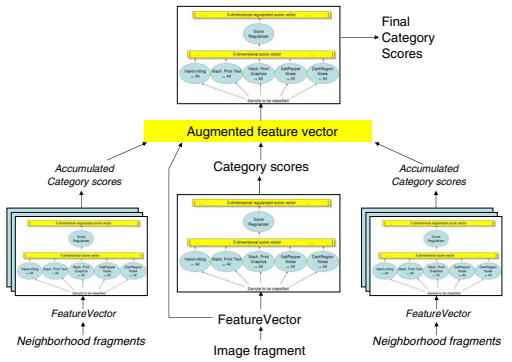


Figure 2. The two stage classifier architecture use for classifying markings. See Figure 3 for an enlarged view of the basic classifier.

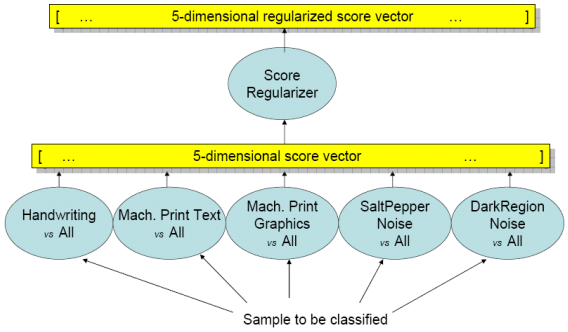


Figure 3. The basic classifier architecture used in each stage of the two stage classification. Each category-vs.-all classifier is learned using AdaBoost.

reported in literature.

Size features: These include width, height, aspect ratio of the bounding boxes, the size of the perimeter, the number of holes in the connected component, the number of foreground pixels. We also include the number of spine-fragments resulting from midcrack thinning, the ratio of foreground count to bounding box area, the ratio of foreground count to the perimeter size, and perimeter size to the bounding box area.

Location features: We measure the minimum horizontal and vertical distances of the fragment from the image boundary. The idea is that this can help to tell shadow noise from dark graphic regions in the document.

Edge Curvature features: The *curvature index* at a contour point is measured as the Euclidean distance of that point from the straight line joining two other contour points that are a fixed contour distance (16 contour points) away from this point. We measure the histogram of curvature indices over each fragment's outer contour.

Contour features: We compute a histogram of relative displacement between contour points that are four contour positions apart. The symmetric and antisymmetric displacement histograms where opposite displacements add and cancel, respectively, are used as features.

Run length features: Spines of fragments are computed by a midcrack thinning algorithm [11]. At each point on the spine we record the minimum and maximum of the horizontal and vertical run-lengths. The histograms of these two numbers are returned as run-length features. Printed parts are expected to have more concentrated run-length histograms than handwriting or noise, though the concentration need not be unimodal.

The next two features are computed from the neighborhood of a fragment.

Regularity features: Machine print is characterized by a high degree of regularity – alignment of location and size. For each fragment, we measure the histogram of differences, in height and location, of neighborhood fragments. These histograms should be highly concentrated (especially around zero) for printed fragments.

Text-line alignment features: Crest and bottom points of the outer contours of fragments are collected, and a RANSAC algorithm is used to find groups of crests, and groups of bottoms that align very well in near-horizontal straight lines. For each fragment we record the number of crest-alignments, and the number of bottom-alignments as features. These are expected to be high for printed text-lines.

3.3 The two-stage fragment classifier

The classification of fragments according to marking type takes place in two stages. In the first stage, each fragment is classified solely on the basis of features described above. This results in each fragment having a per-category score. We could assign, to each fragment, the category with the highest score and stop here. But the classification can be refined by taking into consideration the surrounding context, and how the spatial neighbors get classified. This is achieved in the second stage (Figure 2).

Secondary features: In addition to all the features used in the first stage, the second stage classifier also has access to likely category labels of neighbors measured as *secondary features*. These are accumulations of first-stage category-scores of all fragments with bounding boxes contained in three spatial neighborhoods of the fragment's bounding box: *horizontal strip* ($\pm 160 \times \pm 16$ pixels), *vertical strip* ($\pm 16 \times \pm 160$ pixels), and *rectangular neighborhood* ($\pm 160 \times \pm 160$ pixels). The neighborhood sizes are chosen to be less than one typical character height (16 pixels) and several character heights (160 pixels) at 300 dpi.

Inter-label dependence: There is a subtle but important difference of purpose between the secondary features and

first-stage features that also consider neighborhood context (e.g., regularity features). The secondary features establish a relationship among category-labels of neighborhood fragments, while the first-stage features measure relationships among fragments and their observable properties. If s_i represents the i^{th} fragment, u_i are the features measured on it, and c_i is its category, then the classifier trained on the first stage features gives us $p(c_i|u_j; j \in \text{neighborhood}(i))$. In contrast, the secondary features enable a dependency of the form $p(c_i|c_j; j \in \text{neighborhood}(i))$ (*inter-label dependence*). Hidden Markov Models, Conditional Random Fields [13], and Markov Random Fields [14] are other computational constructs that address this dependence. Our approach is different. We define a neighborhood for each node (fragment), and allow the fragment label to depend on the neighborhood labels. We guide the pattern of dependence by our choice of neighborhoods, but do not enforce a preconceived form of dependence. Rather the dependence, if significant, is learned from training data; the neighborhood features are made available to the second stage classifier learner and are selected if they are found useful for classification. This formulation also sidesteps loopy message propagation or iterative sampling inference which may have compute-time and convergence problems. [1]

Basic classifier: The same basic classifier architecture is replicated in the two stages of classification. The only difference is that the first stage operates on the fragment features, whereas in the second stage the features are augmented with the secondary features (Figure 2).

The basic classifier (Figure 3) itself has two stages. The first stage consists of an array of one-vs.-all classifiers – one per category, and produces an array of category-scores between -1 and 1. The second stage is called the *score regularizer*. It takes the category score array from the first stage and produces an array of refined scores, informed by the claims of every category.

Each one-vs.-all classifier in the first stage is trained using AdaBoost [5]. The weak-learner finds, for each scalar feature component, the best threshold-test classifier. The best from among all the scalar feature classifiers becomes the next weak-classifier. We run 50 AdaBoost iterations for each category. The resulting classifiers are extremely fast at run time. Typically only one of these category-specific classifiers will return a positive score for a fragment, but occasionally either none or multiple categories may lay claim. Also, in general, the scores of independent discriminative binary classifiers are not necessarily comparable. It is the job of the score regularizer to resolve these issues.

[1] In the second stage, unlike loopy belief propagation, we incorporate information only from within our finite neighborhood horizon. A straight forward multi-stage extension with cascaded repetitions of the second stage would lead us to remote evidence propagation.

Table 1. Pixel-label confusion matrix to analyze errors on test data. The numbers are in thousands, and rounded off.

True label ↓	Assigned Labels					Error	Recall	Total
	MachPrint			Noise				
	HWrit	Graph	Text	Dark	S&P		(%)	
HWrit	1197	60	49	39	52	200	85.70	1397
MPGraph	64	1025	140	161	20	386	72.64	1411
MPText	238	145	13996	9	138	530	96.35	14525
NoiseDark	110	91	31	2544	740	972	72.36	3516
NoiseS&P	76	11	43	197	1298	326	79.93	1624
Unknown	2	4	12	8	4	31	.	31
Error	489	311	275	415	954	2444	.	.
Prec. (%)	70.96	76.74	98.07	85.98	57.62	.	.	.
Total	1687	1335	14271	2959	2252	.	.	22504

4 Evaluation

Our system is implemented in Java. A team of volunteers manually labeled nearly 70 document images from various sources including the British American Tobacco litigation documents, NIST Special Database 6 of hand-filled tax forms, document images from the MARG data set, and a few internal documents with complex handwritten parts. 16 images were used for training, the rest for evaluation.

In the summary confusion matrix (Table 1) the pixel counts are accumulated over all of our test images. The overall confusion (error) rate among the categories is 10.86%. This is the percentage of pixels for which the ground-truth labels differ from the system assigned labels. The two noise categories are, by far, the most frequently confused pair. We found that even human labelers have trouble distinguishing them consistently. If we merge the two noise categories, we have a net pixel error rate of 6.70%. If we further merge the two machine print categories, this drops to 5.4%.

Machine print text is the category that is detected with both the highest precision (fraction of pixels assigned this label that are correct), and the highest recall (fraction of pixels with this ground-truth label that are correctly classified). Discounting noise, handwriting has the lowest precision (70.96%), and machine print graphic has the lowest recall (72.64%).

Recall and precision can be traded off by applying rejection criteria. For example a simple rejection algorithm is to reject a segment classification unless only one category claims it. Recall falls (19.5% pixels rejected), but precision improves for all categories. The net error rates fall to 4.6% (3.74% if we merge the noise categories, 2.5% if we further merge the machine print categories.) Overall error rate improved from 14% to 10.86% when we increased our training set from 13 to 16 documents. We expect further improvement with the inclusion of more training examples.

With our java implementation, we recorded a median time of 3.8 sec for segmentation, and 0.6 sec for feature computation, classification, and IO combined. Our images are mostly 300 dpi scans of letter sized pages, and our pro-

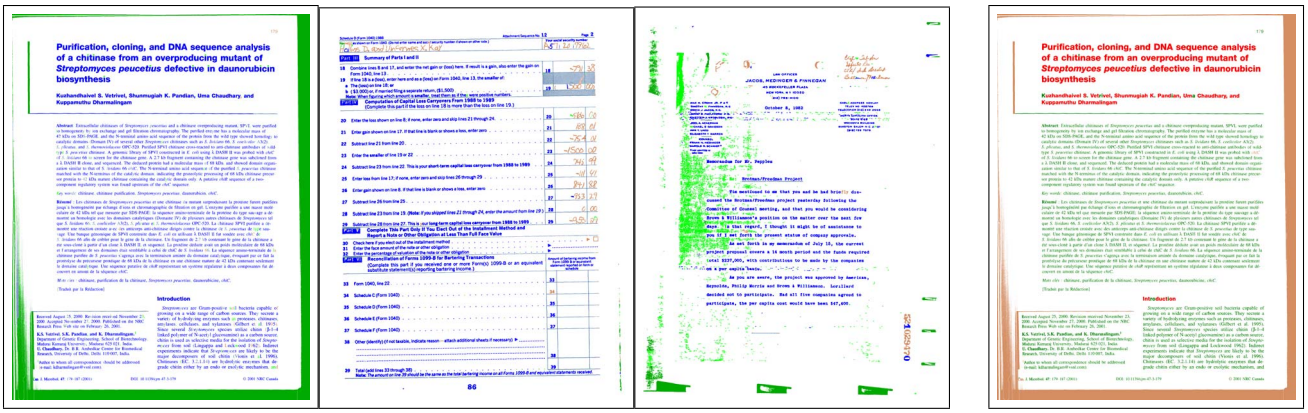


Figure 4. *Left:* **Three examples with difficulty (error rate) increasing from left to right.** *Right:* **System trained to identify title pixels from body-text and noise.**

cessor is 2.7GHz Intel Xeon, with 8GB RAM and 4MB on chip cache. Three examples of results with error rate increasing clockwise from top-left are shown in Figure 4.

5 Discussion and conclusion

The current 93-95% pixel level accuracy that we have been observing (discounting inter-noise confusions) include results on extremely noisy pages. Structures not seen or inadequate in training images (such as stamps, and block graphics) seem to be a major source of errors. It is also apparent that better segmentation (e.g., avoiding oversegmentation) can help avoid many errors. After all, little fragments generated from different marking types are hard to distinguish, and this affects not just classification but also the learning algorithm. Thus knowing when to split, and when not, would be a key enabler of even better performance. It may be possible to learn this from data to replace our current split-until-small-enough algorithm. Segmentation and recognition proceeding in lock-step while informing each other may be useful. Finally, all improvements notwithstanding, it will be necessary to be able to read and recognize familiar shapes to resolve the most difficult cases.

Our solution architecture provides us with a generic framework for a trainable marking categorizer. The rightmost image in Figure 4 shows the output of the same system applied to distinguish headings from body-text and noise.

Acknowledgments Pixel level ground-truth made this work possible. Alex Brito, Doron Kletter, and Subhrendu Sarkar contributed their time to create ground-truth images. We are grateful to them.

References

[1] C. An, H. Baird, and P. Xiu. Iterated document content classification. In *Intnl. Conf. Document Analysis & Recognition*, volume 1, pages 252–256, Los Alamitos, CA, USA, 2007.

[2] D. Bloomberg. Segmentation of handwriting and machine printed text, 1993. US Patent 5181255.

[3] T. Breuel. Segmentation of handprinted letter strings using a dynamic programming algorithm. In *Intnl. Conf. Document Analysis & Recognition*, pages 821–826, 2001.

[4] J. Chen, E. Saund, and Y. Wang. Image objects and multi-scale features for annotation detection. In *Intnl. Conf. on Pattern Recognition*, Tampa Bay, Florida, 2008.

[5] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Eur. Conf. on Comput. Learning Theory*, pages 23–37, 1995.

[6] E. Kavallieratou and S. Stamatatos. Discrimination of machine-printed from handwritten text using simple structural characteristics. In *17th Intnl. Conf. Pattern Recognition*, pages 437–440, Washington, DC, USA, 2004.

[7] K.C.Fan, L.S.Wang, and Y.T.Tu. Classification of machine-printed and handwritten texts using character block layout variance. *Pattern Recognition*, 31(9):1275–1284, 1998.

[8] J. Koyama, A. Hirose, and M. Kato. Local-spectrum-based distinction between handwritten and machine-printed characters. In *Proc. IEEE Intnl. Conf. on Image Proc.*, San Diego, California, October 2008.

[9] J. Liang and R. Haralick. Document image restoration using binary morphological filters. In *Proc. SPIE Conf. Document Recognition*, pages 274–285, 1996.

[10] U. Pal and B. Chaudhuri. Machine-printed and handwritten text lines identification. *Patt. Rec. Lett.*, 22(3-4):431, 2001.

[11] E. Saund. Method and apparatus for extracting the skeleton of a binary figure by contour-based erosion, 2002. US Patent 6377710.

[12] F. Shafait, D. Keysers, and T. M. Breuel. Pixel-accurate representation and evaluation of page segmentation in document images. In *IAPR Intnl. Conf. on Patt. Recognition*, pages 872–875, Hong Kong, China, August 2006.

[13] S. Shetty, H. Srinivasan, M. Beal, and S. Srihari. Segmentation and labeling of documents using conditional random fields. In X. Lin and B. A. Yanikoglu, editors, *Doc. Rec. and Retrieval XIV*, San Jose, CA, USA, 2007. SPIE.

[14] Y. Zheng, H. Li, and D. S. Doermann. Machine printed text and handwriting identification in noisy document images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(3):337–353, 2004.

[15] J. Zhu and M. Moed. Methods and apparatus for gray image based text identification, 2001. US Patent 6301386.