A Method for Document Zone Content Classification

Yalin Wang
Dept. of Elect. Eng.
Univ. of Washington
Seattle, WA 98195
ylwang@u.washington.edu

Ihsin T. Phillips
Dept. of Comp. Science
Queens College, CUNY
Flusing, NY 11367
yun@image.cs.qc.edu

Robert M. Haralick
The Graduate School
CUNY
New York, NY 10016
haralick@gc.cuny.edu

Abstract

This paper describes an algorithm to classify each given document zone into one of nine classes and provides a protocol for its performance evaluation. The classification scheme uses an optimized binary decision tree and Viterbi algorithm for HMM to find the optimal solution. Our algorithm was trained and tested on a total of 24,177 zones within the 1600 images from UWCDROM III database. Its accuracy rate is 98.45% *with a mean false alarm rate of* 0.50%.

1. Introduction

A technical document often contain various types of zones, such as text, math, figure zones, etc. Each of these zones has its own characteristic features. This paper describes an algorithm which assigns zone classes to zones within a given document image.

In the zone classification, each zone is reduced to a feature factor – a set of measurements of pre-defined properties. The features include the run length mean and variance, spatial mean and variance of black and white pixels, etc. A probabilistic model is used to classify each zone on the basis of its feature vector [5]. We employ a decision tree classifier in the classification process (Section 3). Two methods are used to optimize the decision tree classifier to eliminate the data over-fitting problem. We also incorporate context constraints within neighboring zones for some zones and model zone class context constraints as a Hidden Markov Model and used Viterbi algorithm [10] to obtain optimal classification results.

Our earlier work [11] also uses vectors of 67 features. Our new model uses only 25 (Section 2). The algorithm given in Liang et. al [7] discriminates text-zones and none-text zones based on glyphs widths and heights. The algorithm in Chetverikov *et al.* [4] is based on texture features of zones. Le et. al [6] proposed a text-zone logical labeling to label text-blocks as titles, authors, affiliations and abstracts.

We define a performance criteria to evaluate the performance of our algorithm and we conducted a set of experiments evaluating the results in accordance with the performance criteria. The zones are the zone ground-truth entities from the UWCDROM III document image database [8]. The database includes 1,600 scientific and technical document images with a total of 24,177 zones. The zone classes we consider are text with font size $\leq$ 18pt, text with font size $\geq$ 19pt, math, table, halftone, map/drawing, ruling, logo, and others. Our algorithm accuracy rate is 98.45% and the mean false alarm rate is 0.50% (Section 4).

2. Features for Zone Content Classification

Every zone in the document is a rectangular area. Black pixels are assumed to be foreground and the white pixels are background. For each zone, run length and spatial features are computed for each line along two different canonical directions: horizontal, diagonal. These two directions are shown in Figure 1. In the notations, we use subscript h and d to represent these two directions. When discriminating between foreground and background features is necessary, we use superscript 0 and 1 to represent foreground and background features, respectively. For example, $rlmean_h^0$ represents background run length mean feature computed in horizontal direction. A total of 25 features are computed for each zone. In the following, we describe each feature in detail.

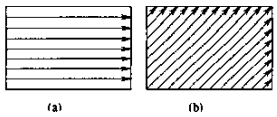

Figure 1. Illustrates the two directions in which we compute run length and spatial features.(a) horizontal; (b) diagonal.

2.1. Run Length Features

A *run length* is a list of contiguous foreground or background pixels in a given direction. A total of 10 run length features are used, they include foreground/background run length mean and variance in each of the two directions.

Let $\mathcal{RL}_h^1$ and $\mathcal{RL}_d^1$ denote the foreground run length sets on the two directions. $|\mathcal{RL}_h^1|$, and $|\mathcal{RL}_d^1|$ constitute the first 2 features.

The next four features include foreground and background run length mean features on two directions in a

1051-4651/02 $17.00 © 2002 IEEE

given zone. Denote them as $rlmean_h^0$, $rlmean_d^0$, $rlmean_h^1$ and $rlmean_d^1$.

$$rlmean_h^0 = \frac{1}{|\mathcal{RL}_h^0|}\sum_{rl\in\mathcal{RL}_h^0} rl, rlmean_d^0 = \frac{1}{|\mathcal{RL}_d^0|}\sum_{rl\in\mathcal{RL}_d^0} rl$$

$$rlmean_h^1 = \frac{1}{|\mathcal{RL}_h^0|}\sum_{rl\in\mathcal{RL}_h^0} rl, rlmean_d^1 = \frac{1}{|\mathcal{RL}_d^0|}\sum_{rl\in\mathcal{RL}_d^0} rl$$

The next four features are foreground and background run length variance features on the two directions in a given zone. Denote them as $rlvar_h^0$, $rlvar_d^0$, $rlvar_h^1$ and $rlvar_d^1$. They are computed as follows.

$$rlvar_h^0 = \frac{\sum_{rl\in\mathcal{RL}_h^0} rl^2}{|\mathcal{RL}_h^0|} - (rlmean_h^0)^2$$

$$rlvar_d^0 = \frac{\sum_{rl\in\mathcal{RL}_d^0} rl^2}{|\mathcal{RL}_d^0|} - (rlmean_d^0)^2$$

$$rlvar_h^1 = \frac{\sum_{rl\in\mathcal{RL}_h^1} rl^2}{|\mathcal{RL}_h^1|} - (rlmean_h^1)^2$$

$$rlvar_d^1 = \frac{\sum_{rl\in\mathcal{RL}_d^1} rl^2}{|\mathcal{RL}_d^1|} - (rlmean_d^1)^2$$

2.2. Spatial Features

Four spatial features are designed to capture the foreground pixel distribution information. We denote the foreground pixel set in a given zone as $\mathcal{F}$. Spatial mean, μ, and spatial variance, δ, can be defined as

$$\mu = \frac{1}{|\mathcal{F}|}\sum_{p\in\mathcal{F}} w_p \qquad \delta = \frac{1}{|\mathcal{F}|}\sum_{p\in\mathcal{F}} (w_p - \mu)^2$$

where w_p is a weight assigned to each foreground pixel p. With two directions, we obtain four features.

As shown in Figure 1, we compute the run lengths from two different directions. In each direction, we start computing from a point on a zone border and continue at a given direction until we hit another zone border again. We call such a computation route a *run segment*. For every run segment the sum of foreground run lengths gives the *run segment projection*. Given a direction, each foreground pixel belongs to only one run segment. We associate each foreground pixel with a weight of run segment projection. We let the foreground pixels in the same run segment have the same weights so we have two different weight definitions according to each direction. We denote the starting and ending pixel coordinates of a horizontal run segment as $(x_{h,1}, y_{h,1})$, $(x_{h,2}, y_{h,2})$. We denote by $(x_{d,1}, y_{d,1})$, $(x_{d,1}, y_{d,1})$ the starting and ending pixel coordinates of a diagonal run segment.

The weights for horizontal and diagonal directions are denoted as w_h and w_d, $w_h = y_{h,1}$ and $w_d = y_{d,2} - x_{d,2}$.

Denote the set of run segments in two directions as $\mathcal{L}_h$ and $\mathcal{L}_d$. For a run segment, say, l_h, we denote its horizontal run segment projection on it as $proj_{h,l}$. In our algorithm, we compute spatial means and spatial variances as follows.

$$spmean_h = \frac{1}{|\mathcal{F}|}\sum_{l\in\mathcal{L}_h} w_h \times proj_{h,l}$$

$$spmean_d = \frac{1}{|\mathcal{F}|}\sum_{l\in\mathcal{L}_d} w_d \times proj_{d,l}$$

$$spvar_h = \frac{1}{|\mathcal{F}|}\sum_{l\in\mathcal{L}_h} [proj_{h,l} \times (w_l - spmean_h)^2]$$

$$spvar_d = \frac{1}{|\mathcal{F}|}\sum_{l\in\mathcal{L}_d} [proj_{d,l} \times (w_d - spmean_d)^2]$$

2.3. Autocorrelation Features

For each run segment, we define four functions: run segment project, number of foreground run lengths, run length mean and spatial mean. We obtain 8 features by computing their autocorrelation functions using Fourier transform.

Denote the set of run length in a horizontal and a diagonal run segment as $\mathcal{RL}_{h,l}$ and $\mathcal{RL}_{d,l}$. The run segment projection function has been defined earlier, $proj_{h,l}$ and $proj_{d,l}$. The function of the number of foreground runs on each run segment are straightforward. The function of run length mean on each run segment can be defined as follows.

$$rlmean_{h,l} = \frac{proj_{h,l}}{|\mathcal{RL}_{h,l}|}, \qquad rlmean_{d,l} = \frac{proj_{d,l}}{|\mathcal{RL}_{d,l}|}.$$

Let $(x_{h,s}, y_{h,s})$, $(x_{h,e}, y_{h,e})$ be the two end points of a horizontal run length, and $(x_{d,s}, y_{d,s})$, $(x_{d,e}, y_{d,e})$ be the two end points of the diagonal run length. The definition of pos and $leng$ functions are given as

$$\begin{aligned} pos_{h,rl} &= x_{h,s}, \qquad leng_{h,rl} = x_{h,e} - x_{h,s} \\ pos_{d,rl} &= x_{d,s}, \qquad leng_{d,rl} = x_{d,e} - x_{d,s} \end{aligned} \tag{5}$$

The spatial mean function for each line can be defined as follows.

$$spmean_{h,l} = \frac{1}{proj_{h,rl}}\Big(\sum_{rl\in\mathcal{RL}_{h,l}} pos_{h,rl} \times leng_{h,rl} + \frac{1}{2}\Big(\sum_{rl\in\mathcal{RL}_{h,l}} (leng_{h,rl})^2 - proj_{h,rl}\Big)\Big)$$

$$spmean_{r,l} = \frac{1}{proj_{d,rl}}\Big(\sum_{rl\in\mathcal{RL}_{d,l}} pos_{d,rl} \times leng_{d,rl} + \frac{1}{2}\Big(\sum_{rl\in\mathcal{RL}_{d,l}} (leng_{d,rl})^2 - proj_{d,rl}\Big)\Big)$$

After we compute one function on each run segment, we can get a sequence of values, indexed by the run segment number. Using the Fast Fourier Transform [9], we can get the autocorrelation functions value for every function. Each feature is the slope of the tangent to the autocorrelation function values whose indexes are close to 0. We used general linear least squares method [9] to compute the slope of the points near 0.

2.4. Background Features

Although some background analysis techniques can be found in the literature([1],[2]), none of them, to our knowledge, has extensively studied the statistical characteristics

of their background structure. Our signature-like background features are designed to give us more information on the distributions of the big foreground chunks in a given zone. We define *large horizontal blank block* and *large vertical blank block* as in [12]. The background feature is the total area of large horizontal and large vertical blank blocks, A.

2.5. Text Glyph Feature

Most of zones have some text glyphs. The information of how many text glyphs a given zone has is also a useful feature. The number of text glyphs in this zone, W, normalized by the zone area is the text glyph feature.

The so-called text glyphs are not from any OCR output. They are outputs of a statistical glyph filter. The inputs of this filter are the glyphs after finding connected component operation. The statistical glyph filter classifies each connected component into one of two classes: text glyph and non-text glyph. The filter uses a statistical method to classify glyphs and was extensively trained on UWCDROM III document image database.

2.6. Column Width Ratio Feature

It is a common observation that math zones and figure zones have a smaller width compared to text zones. For any zone, the quotient of the zone width to the width of its column is calculated as $\frac{C}{Width_{column}}$, where C is the zone width and $Width_{column}$ is the width of the text column in which the zone is.

3. Classification Process

A decision tree classifier makes the assignment through a hierarchical, tree-like decision procedure. For the construction of a decision tree [5], we need a training set of feature vectors with true class labels. At each node, the discriminant function splits the training subset into two subsets and generates child nodes. A discriminant threshold is chosen at each node such that it minimizes an impurity value of the distribution mode at that node. The process is repeated at each newly generated child node until a stopping condition is satisfied and the node is declared as a leaf node on a majority vote.

In building a decision tree classifier, there is a risk of memorizing the training data, in the sense that nodes near the bottom of the tree represent the noise in the sample, As mentioned in [3], some methods were employed to make better classification. We used two methods [12] to eliminate data over-fitting in decision tree classifier.

To further improve the zone classification result, we make use of context constraint in some zone set. We model context constraint as a Markov Chain and use the Viterbi algorithm([10]) to find the most likely state sequence. To apply the Viterbi algorithm([10]), we have to know the probability that each zone belongs to each class. This probability is readily estimated from the training data set by decision tree structure. The details can be found in [12].

4. Experiments and Results

A hold-out method is used for the error estimation in our experiment. We divided the data set into 9 parts. We trained the decision tree on the first 4 parts, pruned the tree using another 4 parts, and then tested on the last 1 part. To train the Markov model, we trained on the first 8 parts and tested it on the last 1 part. This is repeated 9 times. Then the combined 9 part results are put together to estimate the total error rate [5].

True Class	Assigned Class	
	a	b
a	P_{aa}	P_{ab}
b	P_{ba}	P_{bb}

Table 1. Possible true- and detected-state combination for two classes

The output of the decision tree is compared with the zone labels from the ground truth in order to evaluate the performance of the algorithm. A contingency table is computed to indicate the number of zones of a particular class label that are identified as members of one of the nine classes. The rows of the contingency table represent the true classes and the columns represent the assigned classes. We compute four rates here: *Correct Recognition Rate (CR), Misrecognition Rate (MR), False Alarm Rate (FR), Accuracy Rate (AR).* Suppose we only have two classes: a and b. The possible true- and detected-state combination is shown in Table 1. We compute the four rates for class a as follows:

$$CR = \frac{P_{aa}}{P_{aa} + P_{ab}}, MR = \frac{P_{ab}}{P_{aa} + P_{ab}}$$

$$FR = \frac{P_{ba}}{P_{ba} + P_{bb}}, AR = \frac{P_{aa} + P_{bb}}{P_{aa} + P_{ab} + P_{bb} + P_{ba}}$$

Our algorithm was trained and tested on a total of 24,177 zones within the 1600 images from UWCDROM III database. Each zone in the databases belonged to nine different classes: 2 text classes of font size $4 - 18$pt and font size $19 - 32$pt), math, table, halftone, map/drawing, ruling, logo and others. For a total of $24,177$ zones, the accuracy rate was 98.45% and mean false alarm rate was 0.50%, as shown in Table 2.

In Figures 2 and 3, we show some failed cases of our algorithm. Figure 2(a) is a Table zone misclassified as Math zone due to the presence of many numerals and operators. Figure 2(b) is a Map/Drawing zone misclassified as Table zone in that the content of the figure is just a table. Figure 3(a) shows a most frequent error of our current algorithm. Our algorithm classified a Math zone into Text 1 zone class. Sometimes the algorithm still lacks a good ability to detect such a single line math equation zone which, even worse sometimes, includes some description words. Figure 3(b) shows an error example in which a Math zone was misclassified as a table zone because of its sparse nature.

5. Conclusion

Given the segmented document zones, correctly determining the zone content class is very important for the fur-

	T1	T2	M	T	H	M/D	R	L	O	CR	MR
T1	21426	23	40	7	1	7	1	3	3	99.60%	0.40%
T2	19	104	1	0	1	2	0	0	1	81.25%	18.75%
M	47	1	686	2	0	18	1	1	2	90.50%	9.50%
T	6	0	4	162	0	35	0	1	2	77.14%	22.86%
H	1	0	1	1	345	27	0	0	0	92.00%	8.00%
M/D	2	3	20	20	28	648	1	1	5	89.01%	10.99%
R	3	0	2	0	0	2	424	0	1	98.15%	1.85%
L	7	3	1	0	0	0	0	2	0	15.38%	84.62%
O	4	0	2	0	2	7	1	0	6	27.27%	72.73%
FR	3.34%	0.12%	0.30%	0.13%	0.13%	0.42%	0.02%	0.02%	0.06%		

Table 2. Contingency table showing the number of zones of a particular class that are assigned as members of each possible zone class in UWCDROM III. In the table, T_1, T_2, M, T, H, MD, R, L, O represent text with font size $\leq$ 18pt., text with font size $\geq$ 19pt., math, table, halftone, map/drawing zone, ruling, logo, others, respectively.

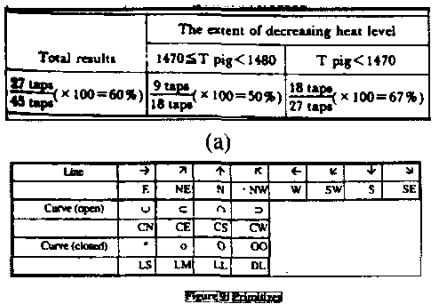

Figure 2. Illustrates some failed examples. (a). Table zone misclassified as Math zone; (b). Map/drawing zone misclassified as Table zone.

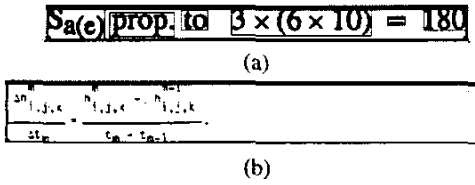

Figure 3. Illustrates some failed examples. (a). Math zone misclassified as Text 1 zone; (b). Math zone misclassified as Table zone.

ther processes. We used only 25 features. We trained and pruned the decision tree. We modeled context constraints as HMM in some zone set. In 1,600 UWCDROM III images, our zone classification method can classify each given zone into one of nine classes. Compared with our previous work [12], the accuracy rate is compatible and the false alarm rate was reduced from 0.53% to 0.50%. Since we used 25 instead 69 features, the classification speed was much improved.

We also showed some failed cases. Many errors are due to the difficult discrimination between single line math and text 1 class. Our future work will include the development of math zone identification technique, modeling zone content dependency feature in a more general zone set.

References

[1] A. Antonacopoulos. Page segmentation using the description of the background. *Computer Vision and Image Understanding*, pages 350–369, June 1998.

[2] H. S. Baird. Background structure in document images. *Document Image Analysis*, pages 17–34, 1994.

[3] W. Buntine. Learning classification trees. *Statistics and Computing journal*, pages 63–76, 1992.

[4] D. Chetverikov, J. Liang, J. Komuves, and R. Haralick. Zone classification using texture features. *Proc. International Conference on Pattern Recognition*, pages 676–680, 1996.

[5] R. Haralick and L. Shapiro. *Computer and Robot Vision*, volume 1. Addison Wesley, 1997.

[6] D. X. Le, J. Kim, G. Pearson, and G. R. Thom. Automated labeling of zones from scanned documents. *Proceedings SDIUT99*, 1999.

[7] J. Liang, R. Haralick, and I. T. Phillips. Document zone classification using sizes of connected components. *Document Recognition III, SPIE'96*, pages 150–157, 1996.

[8] I. Phillips. Users' reference manual. *CD-ROM, UW-III Document Image Database-III*, 1995.

[9] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes in C*. Cambridge University Press, 1988.

[10] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77:257–285, February 1989.

[11] R. Sivaramakrishnan, I. Phillips, J. Ha, S. Subramanium, and R. Haralick. Zone classification in a document using the method of feature vector generation. *Proceedings of the 3rd ICDAR*, pages 541–544, August 1995.

[12] Y. Wang, R. Haralick, and I. T. Phillips. Zone content classification and its performance evaluation. *Sixth International Conference on Document Analysis and Recognition(ICDAR01)*, pages 540–544, September 2001.