Document Classification using Layout Analysis

Jianying Hu   Ramanujan Kashi   Gordon Wilfong
Lucent Technologies Bell Labs
700 Mountain Avenue, Murray Hill, NJ 07974-0636, USA
{jianhu | ramanuja | gtw}@research.bell-labs.com

Abstract

This paper describes methods for document image classification at the spatial layout level. The goal is to develop fast algorithms for initial document type classification without OCR, which can then be verified using more elaborate methods based on more detailed geometric and syntactic models. A novel feature set called interval encoding is introduced to capture elements of spatial layout. This feature set encodes region layout information in fixed-length vectors by capturing structural characteristics of the image. We demonstrate the usefulness of these features derived from interval coding, in a hidden Markov model based page layout classification system that is trainable and extendible.

1 Introduction

In this paper we study features and algorithms for document image classification at the spatial layout level. Document classification has a number of applications such as document organization, retrieval, routing and understanding. We show that an efficient initial classification can be achieved using only layout information. For example, if an incoming fax page is classified as a potential business letter, more detailed logical analysis combining geometric features and text cues derived from OCR can then be applied to confirm the hypothesis and assign logical labels to various fields such as sender address, subject, date, etc., which can then be used for routing, content-based retrieval or other applications [2, 10, 12]).

Our classification scheme relies on features that capture spatial layout similarity. Measuring spatial layout similarity is in general difficult, as it requires characterization of similar shapes while allowing for variations originating both from design and from imprecision in the low level segmentation process. Zhu and Syeda-Mahmood viewed document layout similarity as a special case of regional layout similarity of general images and proposed a region topology-based shape formalism called constrained affine shape model [13]. They use an object (region) based approach and attempt to find correspondences, under constrained affine transforms, between regions in different images based on both shape measures of individual regions and spatial layout characteristics. The drawback of this approach is that it depends on low-level segmentation, which may not always be reliable. In the case of document page segmentation, split or merged blocks are fairly common, especially under noisy conditions.

In this paper layout classification is performed using a trainable statistical classifier in two stages. In the first stage, features that capture layout information are extracted. We use a novel spatial layout representation called *interval encoding*, that can be viewed as an intermediate level of representation. It encodes region layout information in fixed-length vectors, thus capturing structural characteristics of the image while maintaining flexibility. These fixed-length vectors are then clustered and used as features or observation vectors for the second stage that consists of a Hidden Markov Model scheme that is trainable and extendible. For this study we consider five document types which includes magazine pages, 2-column letters, 1-column letters, 2-column journal articles and 1-column journal articles.

2 Page layout feature extraction

As input to our algorithm, we assume that the document page has been deskewed and segmented into rectangular blocks of text [1], as shown in Figure 1. Next the block segmented image is partitioned into an m by n grid and we refer to each cell of the resulting grid as a *bin*. A bin is defined to be a *text bin* if at least half of its area overlaps a single text block and otherwise it is a *white space bin*. In general, we think of a page layout as m rows of n bins where each bin is either labeled as a text bin or a white space bin. We let r_i denote the i^{th} row and index the bins in r_i as $1, \ldots, n$ from left to right. Now our goal is to extract features for comparing page layouts. This comparison is based on a measure of distance between rows on one page with rows on the other page.

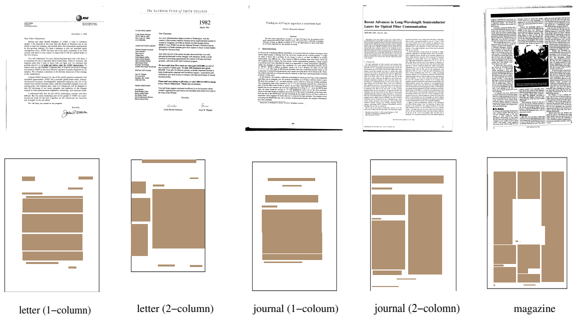

Figure 1. Layout structures and block segmentation of different types of documents.

2.1 Distance computation

The distance between two rows can be computed in numerous ways. The most natural method is based on the the following representation of a row of bins. We define a *block* in a row to be a maximal consecutive sequence of text bins. Suppose the i^{th} row, r_i, has k_i different blocks say, $B^i_1, \ldots, B^i_{k_i}$, ordered from left to right. The bins within a particular block B^i_j are consecutive and we let s^i_j and f^i_j denote the index of the first (leftmost) and last (rightmost) bins in B^i_j respectively. We can represent r_i as a sequence of the pairs $R_i = (s^i_1, f^i_1), \ldots, (s^i_{k_i}, f^i_{k_i})$ and we call such a representation a *block sequence*. Then the distance $d^E(R_i, R_j)$ between two such block sequences R_i and R_j is defined to be the edit distance (see [7]) between them where the cost of inserting or deleting a pair (s, f) is just $f - s + 1$ (the width of the block), and the cost of substituting (s, f) with (u, v) is taken to be $|s - u| + |f - v|$. This distance measure will be referred to as the *edit distance*. While the edit distance appears to be the measure that most accurately captures the differences between rows, it can be computationally unattractive if the lengths of the R_i's become too large. The edit distance also has a disadvantage when it comes to clustering. In particular, given a collection of block sequences that form a cluster it is desirable to be able to compute a cluster center, that is, determine a block sequence R that minimizes the sum of the distances from R to each of the block sequences in the cluster. However it is not at all obvious how such a cluster center should be computed. This leaves the unsatisfactory option of choosing as the cluster center one of the block sequences in the given collection that minimizes the sum of the distances to the others.

We introduce a computationally simpler method that overcomes the clustering difficulty of the edit distance. The scheme is based on the Manhattan distance; that is, the l_1 distance where for two vectors $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ the Manhattan distance between them is given by $l_1(x, y) = \sum_{i=1}^{n} |x_i - y_i|$. In order to use the Manhattan distance we need to represent each row as a fixed length vector. In fact, each row, r_i, will be represented by a vector of length n $(r_i[1], \ldots, r_i[n])$ where $r_i[j]$ is defined as follows. Define $r_i[j] = 0$ if $j \notin [s^i_t, f^i_t]$ for any $t, 1 \leq t \leq k_i$; otherwise $r_i[j] = min\{j - s^i_t + 1, f^i_t - j + 1\}$. We call such a vector, an *interval encoding*. Intuitively, $r_i[j]$ represents how far away in r_i the j^{th} bin is from a white space bin. This gives us a fixed length representation that encodes both the position and the width of the individual block intersections with the i^{th} row. The method defined by computing the Manhattan distance between interval encodings will be called the *interval distance* and will be denoted by d^I.

The effectiveness of interval distance is demonstrated using the examples in Figure 2. Four rows are shown and there are eleven bins per row. The text bins are shown as black and the white space bins are shown as white. For $1 \leq i \leq 4$, the interval encoding $E(row_i)$ of row_i in Figure 2 are given by

$$
\begin{aligned}
E(row_1) &= (1, 2, 3, 4, 3, 2, 1, 0, 1, 2, 1)\\
E(row_2) &= (1, 2, 3, 3, 2, 1, 0, 1, 2, 2, 1)\\
E(row_3) &= (1, 2, 3, 4, 5, 6, 5, 4, 3, 2, 1)\\
E(row_4) &= (1, 0, 1, 2, 3, 4, 5, 4, 3, 2, 1).
\end{aligned}
$$

In the case shown in the figure, row_1 and row_2 are more similar to each other than either is to row_3 since both row_1 and row_2 intersect two blocks at roughly the same horizontal positions whereas row_3 intersects only one very large block. This is reflected by the fact that the edit distance gives $d^E(row_1, row_2) = 2$, much smaller than either $d^E(row_1, row_3) = 7$ or $d^E(row_2, row_3) = 9$. Using the interval distance the results again agrees with perception: $d^I(row_1, row_2) = 6$, while $d^I(row_1, row_3) = 16$ and $d^I(row_2, row_3) = 18$. However, if we simply use d^M, the Manhattan distance between straightforward bitmap representations (i.e., set $r_i[j]$ to 1 if it is a text bin and 0 otherwise), we get $d^M(row_1, row_2) = 2$, larger than $d^M(row_1, row_3) = d^M(row_2, row_3) = 1$. Similarly, row_1 and row_2 are more similar to each other than either is to row_4. Again, the edit distance and interval distance agree with this sense of similarity since $d^E(row_1, row_4) = d^E(row_2, row_4) = 10 > d^E(row_1, row_2) = 2$ and $d^I(row_1, row_4) = 18 > d^I(row_2, row_4) = 17 > d^I(row_1, row_2) = 6$. However, using d^M says that row_1, row_2 and row_3 are each at distance 2 from one another.

Given a collection $E = \{e_1, \ldots, e_t\}$ of interval encodings, we now discuss how they can be partitioned into clusters whose cluster centers can be easily computed. The interval encodings are vectors of length n. A simple and effective means for computing clusters is the k-means clustering technique [4]. This iterative technique requires computing cluster centers for each candidate cluster and for each e_i computing the nearest cluster center to e_i. We define the Manhattan distance to be the distance measure used for

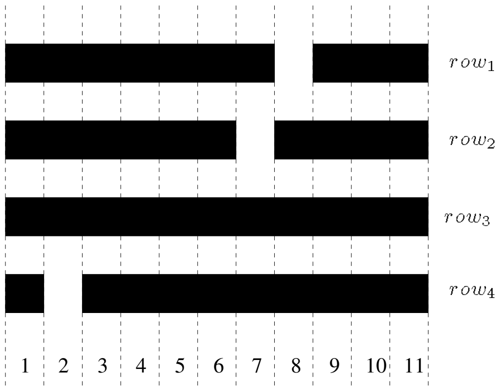

Figure 2. Four examples of rows.

these computations. In this case, a cluster center c is defined to be an n-dimensional vector that minimizes the sum of the Manhattan distances from c to each of the elements of the cluster. It is easy to see then that for n-dimensional sets C, the cluster center c of C is an n-dimensional vector whose i^{th} component is the median of the i^{th} component of the elements of C. Note that in general, c will not necessarily be an interval encoding. Thus unlike in the case of the edit distance described above, we can compute a true cluster center since we are not restricted to choosing some element of a cluster to act as the cluster center.

The above discussion shows that given a collection of interval encodings we can use k-means clustering (based on the Manhattan distance measure) to compute clusters and their centers. This leads to a third method for computing the distance between two rows based on their interval encodings. The distance $d^C(r_i, r_j)$ between interval encodings r_i and r_j is defined to be the l_1 distance between c_1 and c_2 where c_1 is the cluster center nearest (in the l_1 sense) to r_i and c_2 is the cluster center nearest to r_j. We will call this distance measure the *cluster distance*. The computed clusters are then used as feature vectors or observation vectors in the HMM framework discussed in the next section.

3 Hidden Markov Model for classification

We chose to use Hidden Markov Models (HMM) to model different classes of layout designs for several reasons. First, from our observation, a particular layout type is best characterized as a sequence of vertical regions, each region having more or less consistent horizontal layout features (i.e., number of columns and widths and position of each column) and a variable height. Thus a top-to-bottom sequential HMM, where the observations are the interval encoding based cluster features described in the previous section and the states correspond to the vertical regions, seems to be the ideal choice of model. Second, HMMs

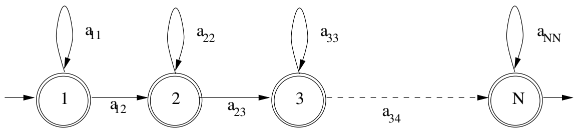

Figure 3. Top-to-bottom HMM state-transition diagram.

have been well studied and have proven to be very effective in modeling other stochastic sequences of signals such as speech or on-line handwriting [9, 5]. Because of the probabilistic nature, they are robust to noise and thus are well suited to handle the inevitable inconsistencies in low-level page segmentation (e.g., vertical splitting or merging of blocks). Furthermore, there exist well established efficient algorithms for both model training and classification [9].

3.1 Model Descriptions

The state transition diagram of a top-to-bottom sequential HMM in the classic form is shown in Fig. 3 (it is drawn left to right here for convenience). A discrete model with N states and M distinct observation symbols $v_1, v_2, \ldots, v_M$ is described by the state transition probability matrix $\mathbf{A} = [a_{ij}]_{N,N}$, where $a_{ij} = 0$ for $j > i + 1$, state conditional observation probabilities: $b_j(k) = Pr(O_t = v_k | s_t = j), k = 1 \ldots M$, and initial state distribution: $Pr(s_1 = 1) = 1$, $Pr(s_1 = i) = 0$ for $i > 1$. We use cluster labels as observations and thus the number of distinct observation symbols M equals the number of clusters, which roughly represents the number of distinct horizontal layout patterns seen in the training set. The number of states should correspond to the maximum number of major vertical regions in a class, and should ideally vary from class to class. However for our initial experiments we used a fixed number of 20 states for all classes. Currently M is chosen empirically to be 30.

One constraint of the classic HMM described above is that state duration (state-holding time) distribution always assumes an exponential from. It is easy to verify that the probability of having d consecutive observations in state i, is implicitly defined as:

$$p_i(d) = (a_{ii})^{d-1}(1 - a_{ii}). \quad (1)$$

In our modeling scheme where states correspond to vertical layout regions and each region may assume a range of possible lengths with similar probabilities, this distribution is inappropriate. In order to allow more flexibility we chose to use a mechanism called explicit duration modeling, resulting in a variable-duration HMM which was first introduced in speech recognition [3] [8] and later proved effective for signature verification as well [6]. This model can also be called a Hidden Semi-Markov Model (HSMM), because the underlying process is a semi-Markov process [11]. In this approach, state-duration distribution $p_i(d)$ can be a general discrete probability distribution. It is easy to prove that any variable duration HMM model is equivalent to the so-called canonical HSMM in which $a_{ii} = 0$ [11]. Thus, for the variable-duration HMM there is no need to estimate state transitional probabilities.

The downside of allowing $p_i(d)$ to be any general distribution is that it has many more parameters and thus requires many more training samples for a reliable estimation. To alleviate this problem we impose a simplifying constraint on $p_i(d)$ such that it can only assume a rectangular shape. Under this constraint, only the duration boundaries τ_i and T_i ($\tau_i < T_i$) are estimated for each state during training. Then the duration probabilities are assigned as: $p_i(d : d > T_i \text{ or } d < \tau_i) = \delta$; $p_i(d : \tau_i \leq d \leq T_i) = (1 - \delta)/(T_i - \tau_i + 1)$, where δ is a small constant. One might point out that with this simplification, we have now replaced one assumption on the shape of p_i (exponential) with another (rectangular). However our experiments show that the latter indeed produces much better results, verifying that it is a justifiable assumption for page layout models.

3.2 Training

We use the Viterbi algorithm for both training and classification as opposed to the more rigorous Baum-Welch and forward-backward procedures because the former can easily accommodate explicit duration modeling without much extra computation [9]. The Viterbi algorithm searches for the most likely state sequence corresponding to the given observation sequence and gives the accumulated likelihood score along this best path. Using explicit state duration modeling, the increment of the log-likelihood score for the transitions to the $i + 1$-st state are given by

$$\Delta L_{i,i+1} = \log b_{i+1}(k) + p_i(d) \tag{2}$$

$$\Delta L_{i+1,i+1} = \log b_{i+1}(k) \tag{3}$$

It should be pointed out that with explicit duration modeling the Viterbi algorithm is no longer guaranteed to find the optimal state sequence, because now the accumulated score of a sequence leading to state i not only depends on the previous state, but also on *how* the previous state was reached (history reflected in the duration d). This dependence violates the basic condition for the Viterbi algorithm to yield optimum solution. However, our experiments showed that the gain obtained by incorporating explicit duration modeling by far outweighs the loss in the optimality of the algorithm.

The HMM of each layout class is trained by applying a segmental k-means iterative procedure [9] across all the training samples in the class. The procedure is composed of iterations of the following 2 steps:

1. Segmentation of each training sample by the Viterbi algorithm, using the current model parameters.

2. Parameter re-estimation using their means along the path.

The initial model parameters are obtained through equal-length segmentation of all the training samples. The re-estimation stops when the difference between the likelihood scores of the current iteration and those of the previous one is smaller then a threshold (usually after 3-5 iterations).

3.3 Experiments

Experiments were conducted to classify a test document into one of five classes. During classification, first preprocessing and feature extraction are carried out and the observation sequence is computed for a given page. Then the Viterbi algorithm is applied to find the likelihood of the observation sequence being generated by each one of the K class models. The page is then labeled as a member of the class with highest likelihood score. Alternatively, more than one class with highest likelihood score can be retained as potential class labels to be examined more carefully later on. The five classes of documents used in the study were one-column journal articles (ICJ), two-column journal articles (2CJ), one-column letters (1CL), two-column letters (2CL) and magazine articles (M). Thirty samples in each class were used to build a Hidden Markov model template for that class. A test of eighty nine monochrome document samples at 300 dpi were used in the classification experiment. Each test sample was then compared with the five models to generate likelihood scores. The results of the classification for the test samples is shown in Table 1. Each row of the table corresponds to one of the five classes of documents.

As seen in Table 1, sixteen of the 1-column-journals obtained the highest likelihood scores when compared against the template of 1-column journal class and hence were correctly classified. Six of the test samples which were 1-column journal articles were incorrectly classified as 1-column letters. A closer look at the two document classes, 1-column journal articles and 1-column letter articles, reveals that their spatial layouts are quite similar. Differences between these two classes are more apparent at the top of the page. In letters, the top part of the page has address blocks and in journals, it is usually a title with the author information. Thus accurate discrimination between these two classes requires more detailed geometrical/syntactical analysis incorporating OCR results. Similar observations occur in the third row showing the results of classification

	1CJ	2CJ	1CL	2CL	M
1CJ	16	0	6	0	0
2CJ	0	18	0	0	0
1CL	3	0	7	0	0
2CL	0	2	0	7	2
M	0	4	0	2	24

Table 1. Classification results on the five classes of documents

	$Accuracy_1$	$Accuracy_2$
1CJ	73	100
2CJ	100	100
1CL	70	100
2CL	64	90
M	86	100

Table 2. Percentage accuracy in classification of test documents.

with one-column letters. In this case seven of the test documents were classified correctly and three of the test samples had the highest likelihood score when compared with the template of a one-column journal article.

The accuracy for each of the five classes of documents is shown in Figure 2. As explained earlier, in the classification experiment, each test sample was compared with five models (classes). $Accuracy_1$ refers to the percentage of documents whose correct class appeared as the top-choice in the ranking experiment (top-one accuracy). $Accuracy_2$ refers to the percentage of documents whose correct class appeared in either the top or the second choice (top-two accuracy).

As seen from Table 2, good accuracy scores are achieved in classification based solely on spatial characteristics and without OCR. In particular, the top-two accuracy scores are very high for all but one class, demonstrating the effectiveness of the method as an initial "screening" stage. The accuracies for the two-column letter class are relatively low. One possible reason is that the samples we have collected for that class include many commercial "junk mails" with highly varied layouts and fancy graphic items, causing more errors in segmentation. As a result, the model for that class is relatively weak and thus the two-column letter test samples are more likely to be mis-classified.

4 Conclusions

This paper describes algorithms for document classification based on information related to spatial layout. Using a novel feature called interval encoding, experiments were conducted to classify pages in terms of document types in an HMM framework. As seen from the results of the classification experiments, reasonably high accuracy scores are obtained based solely on spatial layout and without OCR. These experiments demonstrate that the features and methods introduced in this paper provide efficient and effective initial classification of documents. Results from this initial analysis can be used by later stages of document processing, including use of more detailed, class specific geometrical and syntactic models incorporating OCR results, to achieve better classification as well as content understanding and extraction.

Acknowledgments

We would like to thank Henry Baird and John Hobby for the numerous helpful discussions.

References

[1] H. S. Baird. Background structure in document images. *Int Journal of Pattern Recognition and Artificial intelligence*, 8(5):1013–1030, 1994.

[2] A. Dengel and F. Dubiel. Computer understanding of document structure. *Int Journal of Imaging Systems and Technology*, 7:271–278, 1996.

[3] J. D. Ferguson. Variable duration models for speech. In *Proc. Symp. on the Application of HMM to Text and Speech*, pages 143–179, Priceton, NJ, 1980.

[4] A. Gersho and R. M. Gray. *Vector quantization and signal compression*. Kluwer Academic Publishers, 1992.

[5] J. Hu, M. K. Brown, and W. Turin. HMM based on-line handwriting recognition. *IEEE PAMI*, 18(10):1039–1045, Oct. 1996.

[6] R. Kashi, J. Hu, W. Nelson, and W. Turin. On-line handwriting signature verification using hidden Markov model features. In *Proc. ICDAR'97*, Ulm, Germany, August 1997.

[7] J. B. Kruskal and D. Sankoff, editors. *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*. Addison-Wesley, Reading, MA, 1983.

[8] S. E. Levinson. Continuously variable duration hidden Markov models for automatic speech recognition. *Computer Speech & Language*, 1(1):29–45, Mar. 1986.

[9] L. R. Rabiner and B. H. Juang. *Fundamentals of Speech Recognition*. Prentice Hall, Englewood Cliffs, NJ, 1993.

[10] S. L. Taylor, M. Lipshutz, and R. W. Nilson. Classification and functional decomposition of business documents. In *Proc. ICDAR'95*, pages 563–566, Montreal, Canada, August 1995.

[11] W. Turin. *Performance Analysis of Digital Transmission Systems*. Computer Science Press, New York, 1990.

[12] H. Walischewski. Automatic knowledge acquisition for spatial document interpretation. In *ICDAR'97*, pages 243–247, Ulm, Germany, Aug. 1997.

[13] W. Zhu and T. Syeda-Mahmood. Image organization and retrieval using a flexible shape model. In *IEEE Int. Workshop on Content Based Access of Image and Video Databases*, pages 31–39, Bombay, India, Jan. 1998.