UNSUPERVISED STYLE CLASSIFICATION OF DOCUMENT PAGE IMAGES

Song Mao,[a] *Lan Nie,*[b]*, and George R. Thoma*[a]

[a]U.S. National Library of Medicine, Bethesda, MD 20894, USA
[b]Computer Science and Engineering Department, Lehigh University, PA 18015, USA

Style classification of document page images is crucial for logical structure analysis of heterogeneous collections of documents. Both layout and contextual features contain significant information about document styles. Most existing methods are supervised methods in which specific document models or classifiers are learned from a training set of document page images with known class labels. In this paper, we propose an unsupervised classification method that involves no training or manual selection of algorithm parameters. In particular, we first represent each document page as an ordered labeled X-Y tree. A tree matching algorithm is then used to compute style dissimilarity between two document pages. We propose a set of tree edit cost functions based on Karl Pearson distance between two multivariate feature observations, which is robust to the over-segmentation problem and zone length variations of same logical entities. Finally, the K-medoids algorithm is used to find an optimal grouping of the trees into K clusters, each of which corresponds to a distinct document style. We evaluate our algorithm on test datasets with different cluster sizes and degrees of style similarity. Experimental results show our algorithm achieved an average classification accuracy of 95.69% over six datasets consisting of 150 pages of 11 different styles.

1. INTRODUCTION AND PRIOR WORK

The style of a document page is represented by both layout and contextual features of its physical zones. While it is difficult to design a document image analysis algorithm that works for an arbitrary set of documents, domain specific knowledge such as document style is very useful for logical structure analysis of documents, which is essential in a document image analysis system. Document identifiers such as header and footer text could contain significant information (e.g. the name of a journal) about document styles. However, documents from different journal titles could have the same style and vice versa. For example, articles from two different journals can have the same one-column layout and same font size and attribute for the same logical entities, and documents in the same journal could have different article types such as regular paper and correspondence, each of which has a distinct style.

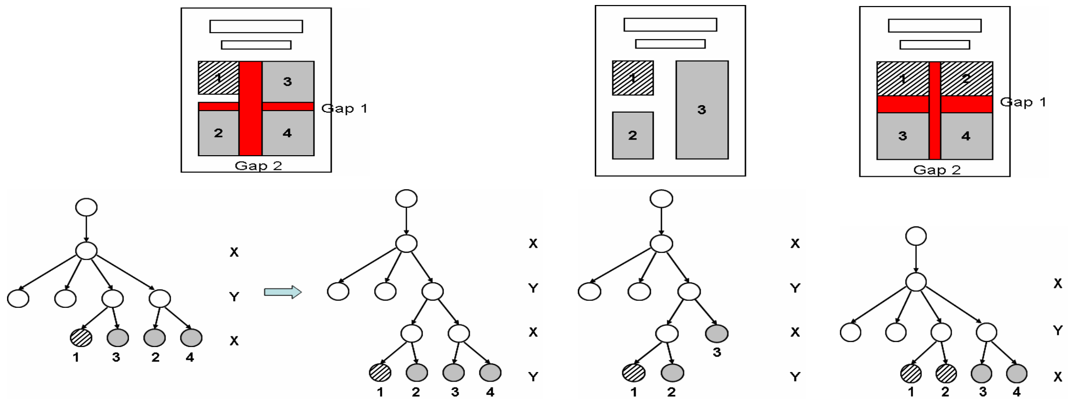
Most document image classification methods follow the supervised approach where classifiers or specific document class models are learned from training data with known class labels. Diligenti *et. al.* [1] propose Hidden Tree Markov Models (HTMMs) for classifying commercial invoices. Cesarini *et. al.* [2] use artificial neural networks to classify documents modeled by a vector representation of modified X-Y trees. Baldi *et. al.* [3] first use tree grammars to expand training sets consisting of modified X-Y trees with class labels, and then use a $K - nn$ approach to classify the pages. Hu *et. al.* [4] propose a novel feature set called interval encoding to encode region layout information, and use it in a hidden semi-Markov model (HSMM) to classify letters and journal pages. All the above methods rely on training of specific document class models or classifiers. When such labeled training data is not available or expensive to create, unsupervised classification methods are necessary.

In this paper, we propose an unsupervised classification method. Character font size and zoning results of a document page are used to build an ordered labeled X-Y tree. A set of edit cost functions are proposed, based on the Karl Pearson distance between two multivariate feature vectors. They are used in a dynamic programming algorithm to compute an edit distance between any pair of such trees. Finally, a K-medoids algorithm is used to find an optimal grouping of these trees into K clusters by minimizing a within-cluster distance measure. Each of the obtained clusters corresponds to a distinct document class.

This paper is organized as follows. In Section 2, we represent each page as an ordered labeled X-Y tree. In Section 3, we provide a detailed description of a tree matching algorithm and associated edit cost functions. The clustering algorithm is presented in Section 4. Finally in Section 5, we describe the experimental protocol, report results, provide a discussion, and give future directions.

2. DOCUMENT PAGE IMAGE REPRESENTATION

A document page image having the Manhattan layout can be represented as an ordered labeled X-Y tree [1, 2, 3, 5] since it concisely summarizes layout and contextual styles of the page. Each node in the tree denotes a document region where root node denotes the whole document page.



(a) (b) (c)

Fig. 1. An ordered labeled X-Y tree is built for each of the three documents shown in (a) (left tree), (b), and (c) using their zone bounding boxes. Page (a) and (b) have the same layout and contextual type, but their trees are quite different. After removing the ambiguity using the widths of Gap 1 and Gap 2, a new tree (right of arrow) is built for page (a), which is similar to tree (b) but is different from tree (c). Note that gray and shaded zones represent different logical entities, numbers 1,2,3,4 represent read order, and notation X and Y denote the projection axes.

A parent-child relationship represents hierarchical containment of the physical regions in a document page. The children nodes of a parent node is read left to right. Each node is labeled with a feature vector, which represents the layout and contextual characteristics of the corresponding region.

2.1. Build Ordered and Labeled Tree

Zone bounding boxes in the whole page are first projected onto the X axis. Subsequently, each non-overlapping zone on a projection profile denotes a leaf node, and overlapping zones are merged into a single node, which will be projected onto a different axis in the next projection step. We repeat this process until all leaf nodes of the tree denote singular zones as shown in Figure 1 (b). Since noise zones and skew can adversely affect this process, document pages should be deskewed and cleaned in a preprocessing step.

Ambiguity could arise in this process in that trees with identical structure may be built for documents with quite different layout and contextual styles. An example is shown by the left tree in Figure 1 (a) and the tree in (c). We can remove the ambiguity by first splitting at the wider gap as shown in the right tree in Figure 1 (a). At tree level 2, we do not split zones at Gap 1 since it is thinner than that the intersecting Gap 2. However in Figure 1 (c), the split at Gap 1 precedes the separation at Gap 2 since Gap 1 is thicker than Gap 2. This is a reasonable operation since document style editors typically use gaps (or field separators) of different widths to signify logical separations at different degrees.

2.2. Feature Selection for Labeling Tree Nodes

Each node v in an ordered labeled tree T is labeled with a 4-tuple feature vector $\mathbf{f}^v = (f_1^v, f_2^v, f_3^v, f_4^v)$, where f_1^v, f_2^v, and f_3^v denote average character font size, level number in T, and center X coordinate (in pixels) of v, respectively. If v is a node on a Y projection profile, f_4^v denotes the minimum vertical distance (in pixels) between v and its top or bottom sibling on the profile. We see that this feature vector contains both the contextual (f_1^v) as well as layout related information (f_2^v, f_3^v, f_4^v). In the next section, we will describe how the feature vector is used to compute an edit distance between a pair of ordered labeled X-Y trees.

3. COMPUTE EDIT DISTANCE BETWEEN TREES

Zhang *et. al.* [6] proposed an efficient algorithm for matching ordered labeled trees using relabel, deletion, and insertion operations on tree nodes. We adapt this algorithm to compute an edit distance between two ordered labeled X-Y trees.

3.1. The Tree Matching Algorithm

Let T be an ordered labeled X-Y tree and let $V(T)$ be the set of nodes in T. Let $T(v)$ be the subtree of T rooted at a node $v \in V(T)$. Let F_1 and F_2 be two ordered forests. Let $D(.,.)$ be the edit distance between its two arguments, which can be either trees or forests. Let C be a cost function defined on labels (feature vectors) of tree nodes. Let r_1 and r_2 be the rightmost roots of the trees in F_1 and F_2. Let ϕ and Φ denotes empty node and empty tree, respectively. The algorithm described in [6] computes a tree edit distance based on the following dynamic programming procedures: $D(\Phi,\Phi) = 0, D(F_1,\Phi) = D(F_1 - r_1,\Phi) + C(r_1 \rightarrow \phi), D(\Phi, F_2) = D(\Phi, F_2 - r_2) + C(\phi \rightarrow r_2)$,

$$D(F_1, F_2) = \min \begin{cases} D(F_1 - r_1, F_2) + C(r_1 \rightarrow \phi), \\ D(F_1, F_2 - r_2) + C(\phi \rightarrow r_2), \\ D(F_1 - T(r_1), F_2 - T(r_2)) + \\ D(T(r_1), T(r_2)), \end{cases}$$

$$D(T(r_1), T(r_2)) = \min \begin{cases} D(T(r_1) - r_1, T(r_2)) + C(r_1 \to \phi), \\ D(T(r_1), T(r_2) - r_2) + C(\phi \to r_2), \\ D(T(r_1) - r_1, T(r_2) - r_2) + C(r_1 \to r_2). \end{cases}$$

Deletion, insertion, and relabel cost functions $C(r_1 \to \phi)$, $C(\phi \to r_2)$, $C(r_1 \to r_2)$ are described in the next section.

3.2. Edit Cost Functions

In the problem of document page style classification, the edit distance between two ordered labeled X-Y trees is used as a dissimilarity measure between the represented document page images. Therefore, the cost functions for node deletion, insertion, and relabel operation should reflect the degree of dissimilarity between a pair of document page images. In order to make the edit distance computation symmetric between two trees, we define identical cost functions for deleting and inserting the same node in a tree.

The cost to relabel a node v in tree T_1 to a node w in tree T_2 is defined in Equation 1 as Karl Person distance (a type of weighted Euclidean distance) between the two multivariate feature vectors, each of which consists of the first three feature variables in $\mathbf{f}^v$ and $\mathbf{f}^w$, respectively.

$$C(v \to w) = C(w \to v) = \sqrt{\sum_{i=1}^{3} \frac{(f_i^v - f_i^w)^2}{s_i^2}}, \qquad (1)$$

where s_i^2 is the variance of feature variable f_i. Here we use sample variance to approximate s_i^2. Since the feature variables have different measurement units, their sample variances are used as weights so that the computed cost is approximately scale-invariant. Similarly, we define the cost for deleting (or inserting) a node v in a tree as follows: If v is obtained from a X projection profile,

$$C(v \to \phi) = C(\phi \to v) = \sqrt{\sum_{i=1}^{3} \frac{(f_i^v - f_i^{'})^2}{s_i^2}}. \qquad (2)$$

If v is obtained from a Y projection profile,

$$C(v \to \phi) = C(\phi \to v) = \sqrt{\sum_{i=1, i \neq 3}^{4} \frac{(f_i^v - f_i^{'})^2}{s_i^2}}, \qquad (3)$$

where $f_i^{'}$ denotes the i^{th} variable of feature vector $\mathbf{f}^{'}$. If v has a left sibling, $\mathbf{f}^{'}$ is the label of that sibling. Otherwise, $\mathbf{f}^{'}$ is the label of the right sibling of v, if v has one. If v does not have any sibling, set $\mathbf{f}^{'} = (0, 0, f_3^v, f_4^v)$, i.e., only the first two features are used to compute a cost to delete or insert a singular child in Equation 2 or 3.

Note that the relabel cost function does not involve the vertical minimum distance features f_4^v and f_4^w. This is because the cost of relabeling one zone into another should be small if they have very similar contextual and layout styles (except their lengths). The deletion (or insertion) cost functions compute a small cost for deleting a part from the whole on the Y projection profile, and therefore is insensitive to the over-segmentation problem in zoning results. On the other hand, the cost of using the center X coordinate feature to penalize the deletion of a column from a multi-column page should be large.

4. TREE CLUSTERING

Combinatorial clustering algorithms directly work on the observations without assuming an underlying probability model. The popular K-means algorithm uses squared Euclidean distance as the dissimilarity measure and requires that observations can be considered as points in the Euclidean space.

In our case, observations are ordered labeled X-Y trees that cannot be readily represented as points in the Euclidean space. On the other hand, we can compute a dissimilarity measure (tree edit distance) for each pair of trees. We cluster our trees using the K-medoids algorithm [7] since it only makes use of the pairwise dissimilarity measure of the observations. Moreover, the expensive tree edit distance computation is a one-time operation and is not involved in the optimization steps of the K-medoids algorithm. Hastie *et. al.* [8] proposed a method based on the Gap statistic to estimate the number of clusters in a set of data, which can be applied to our case. Assuming the number of clusters is K, the detailed steps of the algorithm are as follows:

1. Randomly choose a set of K initial cluster centers $\mathbf{T}^0 = (T_1^0, \ldots, T_K^0)$ from $T_1, \ldots, T_N$. Let $\mathbf{T}^c = \mathbf{T}^0$.
2. For the current K cluster centers $\mathbf{T}^c$, assign each tree to the closest (in terms of tree edit distance) cluster center: $L(i) = \arg\min_{1 \le k \le K} D(T_i, T_k^c), i = 1, \ldots, N$, where $L(i)$ denotes the cluster label of tree T_i.
3. Given cluster assignment L, find a new center in each cluster that minimizes total distance to other points in that cluster: $i_k^* = \arg\min_{i:L(i)=k} \sum_{i':L(i')=k} D(T_i, T_{i'})$. Update current cluster centers as $T_k^c = T_{i_k^*}, k = 1, \ldots, K$.
4. Repeat steps 2 and 3 until assignments do not change. Let L^* and T^{*c} be final assignments and cluster centers. Compute a within-cluster distance measure (corresponding to $\mathbf{T}^0$) as $\sum_{k=1}^{K} \sum_{i:L^*(i)=k} D(T_i, T_k^{*c})$.

5. EXPERIMENTS

We collected 150 article title page images from 9 journal issues. They belong to 11 distinct styles in terms of layout and contextual features as shown in Figure 2. We see they consist of three major layouts: one column, two column, and three column pages. Each major layout contains document styles with different degrees of finer style variation. These document styles or clusters do not all have the same size. The smallest cluster only has three pages and the largest one has 24 pages. Six datasets with different document style and cluster size compositions are selected to evaluate our method.

We run the K-medoids algorithm using 20 randomly selected initial cluster center sets and consider the clustering result giving the minimum within-cluster distance measure as the final result. In the clustering results, each cluster is labeled by the document style that has the maximum number of pages, and pages belonging to other document styles in the cluster are counted as misclassified. Statistics associated with each of the six datasets and their classification results are reported in Table 1. We see that our method works well on both relatively balanced clusters in datasets II, III, IV, V and unbalanced clusters in datasets I and VI. When a dataset is very unbalanced, smaller clusters tend to be merged into larger ones. For example in experimental results on the dataset VI, three pages from style (10) are merged into a larger cluster. Curves in Figure 3 characterize the relationship between minimum within-cluster distance measure and the number of randomly selected initial cluster center sets for all six datasets. We see that the K-medoids algorithm quickly finds an optimal grouping of the document pages in most datasets after using only one or a few randomly generated initial cluster center sets.

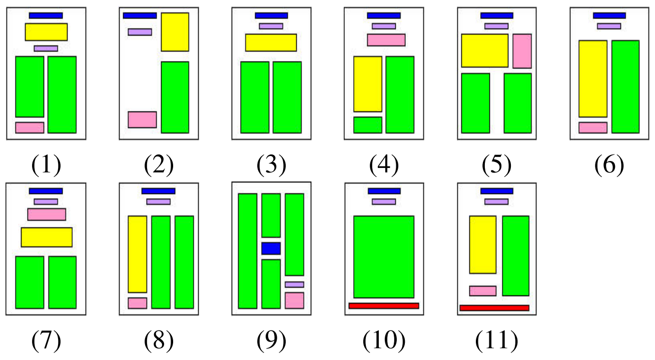


Fig. 2. The 11 document styles in our dataset. Note that different zone shadings represent different logical entities.

Table 1. Dataset descriptions and their classification results. Note that in the second column, we use the document style numbers shown in Figure 2.

Dataset	(Style):size	No. of pages	No. of misclassified pages	Classification accuracy
I	(9):8, (10):3, (11):14.	25	1	96%
II	(1):10, (2):15, (3):9, (4):24, (5):22, (6):16.	96	4	95.83%
III	(8):9, (9):8	17	1	94.12%
IV	(2):15, (5):22, (8):9.	46	0	100%
V	(1):10, (2):15, (3):9, (4):24, (5):22, (6):16, (8):9.	105	4	96.19%
VI	(1):10, (2):15, (3):9, (4):24, (5):22, (6):16, (7):20, (8):9, (9):8, (10):3, (11):14.	150	12	92%

Future work will include clustering data with severely unbalanced clusters, extracting high dimensional features and projecting them onto low dimensional manifold, and estimating cluster number K using the Gap statistic [8].

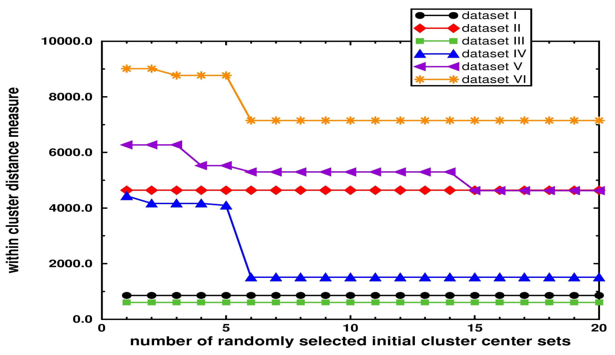


Fig. 3. Relationships between within-cluster distance and number of initial cluster center sets for all 6 datasets.

6. REFERENCES

[1] M. Diligenti, P. Frasconi, and M. Gori, "Hidden tree Markov models for document image classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, pp. 519–523, 2003.

[2] F. Cesarini, M. Lastri, S. Marinai, and G. Soda, "Encoding of modified X-Y trees for document classification," in *Sixth International Conference on Document Analysis and Recognition*, Seattle, Washington, September 2001, pp. 1131–1136.

[3] S. Baldi, S. Marinai, and G. Soda, "Using tree-grammars for training set expansion in page classification," in *Seventh International Conference on Document Analysis and Recognition*, Edinburgh, Scotland, August 2003, pp. 829–833.

[4] J. Hu, R. Kashi, and G. Wilfong, "Document classification using layout analysis," in *10th International Workshop on Databases and Expert Systems Applications*, Florence, Italy, September 1999.

[5] M. Krishnamoorthy, G. Nagy, S. Seth, and M. Viswanathan, "Syntactic segmentation and labeling of digitized pages from technical journals," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, pp. 737–747, 1993.

[6] K. Zhang, D. Shasha, and J. T. L. Wang, "Approximate tree matching in the presence of variable length don't cares," *Journal of Algorithms*, vol. 16, pp. 33–66, 1994.

[7] L. Kaufman and P. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*, Wiley, New York City, New York, 1990.

[8] T Hastie, R. Tibshirani, and G. Walther, "Estimating the number of data clusters via the Gap statistic," Tech. Rep. 208, Stanford University, Stanford, CA, March 2000.