Simultaneous Layout Style and Logical Entity Recognition in a Heterogeneous Collection of Documents

Siyuan Chen, Song Mao, and George R. Thoma
U.S. National Library of Medicine
Bethesda, MD 20894, USA

Abstract

Logical entity recognition in heterogeneous collections of document page images remains a challenging problem since the performance of traditional supervised methods degrade dramatically in case of many distinct layout styles. In this paper we present an unsupervised method where layout style information is explicitly used in both training and recognition phases. We represent the layout style, local features, and logical labels of physical regions of a document compactly by an ordered labeled X-Y tree. Style dissimilarity of two document pages is represented by the distance of their representing trees. During the training phase, document pages with true logical labels in training set are classified into distinct layout styles by unsupervised clustering. During the recognition phase, the layout style and logical entities of an input document are recognized simultaneous by matching the input tree to the trees in closest-matched layout style cluster, of training set. The experimental results show that our algorithm is robust to balanced and unbalanced style cluster sizes, zone over-segmentation, zone length variation, and variation in tree representations of the same layout style.

1. Introduction

Logical structure analysis is often an important goal of a document image analysis system. Recognized logical entities are essential for searching, accessing, indexing, and retrieving the information contained in document images.

Numerous algorithms have been proposed for logical structure analysis. Many methods are based on heuristic rules [6, 4, 11, 10]. Since documents could have different layout styles, a different set of rules will have to be manually created for each such layout style. If the number of distinct layout styles is large, the number of rules that need to be manually created also become large, thereby decreasing the recognition speed. Some researchers represent heuristic rules by deterministic or stochastic formal grammars [7, 12], and use various parsing techniques to recognize logical entities. For each distinct layout style, grammatical rules have to be manually created as well in these methods.

Graph models such as the tree have been used to better represent the two-dimensional arrangement of logical entities on a document page image. Krishnamoorthy et al. [7] represent the logical structure of a document page by an X-Y tree and use block grammars and associated parsing algorithm for logical entity recognition. Dengel and Dubiel [1] use Geometric Tree or GTree and associated parsing algorithm to represent and recognize logical structure of letters in a learning framework.

Liang et al. [8] represent the layout of document pages by a fully connected attributed graph and performed page classification and logical labeling at the same time. Since each node in the model denotes a segmented zone and each edge between two nodes represent spatial relationship between the two zones on a document page, the model cannot represent hierarchical nesting relationship among document regions and is also sensitive to variation in distance between two zones and zone over-segmentation problem. The definition of a match cost makes the search for the minimal cost match of two graph models computationally expensive. In their experiments, layout models for different document classes are manually built, and most data are used in both the training and evaluation phases of their algorithm. Moreover, the forty pages that are only used in the evaluation phase are used to evaluate the performance of logical labeling, not that of page classification.

While our method is conceptually similar to this method, it differs significantly in the layout model representation, model learning in the training phase, and experimental verification. We will show in our experimental results that our algorithm is robust to balanced and unbalanced style cluster sizes in training dataset, variation in distance between zones, zone over-segmentation problem, and zone length variation of same logical entities.

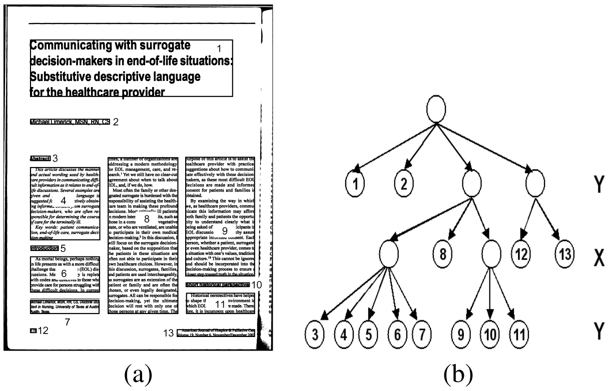
This paper is organized as follows. In Section 2, we represent the layout style, local features, and logical labels of physical regions of a document page by an ordered labeled X-Y tree. In Section 3, the detailed logical entity recognition training and recognition algorithms are given. In Section 4, we present experimental results and discussions. Finally, we summarize our paper in Section 5.

2. Document Page Representation: Ordered Labeled X-Y Tree

It is essential to be able to represent a document page and its associated structural and feature information in a unified and compact form so that rigorous computations can be performed on them. We use an ordered labeled X-Y tree [7] to compactly represent the layout style, local features, and logical labels of physical regions of a document page image that has the Manhattan layout. Each node in the tree represents a physical region or a zone and the root node represents the entire document page. Each leaf node represents a segmented zone and each internal node represents a group of segmented zones in a document page image. An internal node at current tree level is projected into multiple nodes on either the X or Y axis at the next tree level. Hierarchical nesting relationship among different logical entities is represented by parent-child relationship in the tree. The child nodes of a parent node are read from left to right. Each node is also associated with a set of local features of the corresponding physical region. Our basic assumption is that two document page images should have similar ordered labeled X-Y tree representations if they have similar layout styles, local features and logical labels of corresponding physical regions.

We use a tree-matching algorithm [13] to compute an edit distance between two document page images. Note that if two documents have similar layout style but their logical entities have very different local feature values and are placed at different locations the edit distance between the two can still be large. Figure 1 shows a document page and its corresponding ordered labeled X-Y tree representations. This layout style information represented by the ordered labeled X-Y tree will be used explicitly in both training and recognition phases of our algorithm as described in Section 3.

Each node in an ordered labeled tree is associated with a label, which is represented by a set of features extracted from the corresponding document region. Using OCR engine FineReader 8.0 we extracted raw OCR features which include ASCII characters, font features and the bounding box coordinates of zones and characters. For each node, we extracted the following features: average font size, level of the node in the tree, direction of projection profile of the node, the X coordinate of the vertical center line of the corresponding zone bounding box, the distance to its nearest sibling in Y direction, the ratio of the number of digits to the number of characters, the ratio of the number of capital letters to the number of characters, the ratio of the number of letters and digits to the number of characters, the ratio of the number of letters to the number of letters and digits, the ratio of the number of capital letters to the number of letters.


Communicating with surrogate decision-makers in end-of-life situations: Substitutive descriptive language for the healthcare provider
1
2
3
4
5
6
7
8
9
10
11
12
13
Y
X
Y

(a) (b)

Figure 1. A document page (a) and its ordered labeled X-Y tree representation (b). Each leaf node is associated with a segmented zone on the page (a) and a set of local features (which are not shown here) extracted from the zone.

3. Algorithms

3.1. Tree Matching by Dynamic Programming

In Zhang's paper [13], a dynamic programming algorithm was proposed to generate an optimal matching between two ordered trees in terms of minimum edit distance. Three types of edit operations: relabeling, deletion and insertion are used to describe the sequence of operations to transform the data tree T_D into the reference tree T_R. The mapping is isomorphic and sibling and ancestor orders are preserved. The algorithm runs in $O(|T_R| \times |T_D| \times \min(depth(T_R), leaves(T_R)) \times \min(depth(T_D), leaves(T_D)))$ time, where $|T_R|$ denotes the total number of nodes in tree T_R.

3.1.1 Formulation of Edit Distance Function

Let the cost function of edit operation be $\gamma(u \rightarrow v)$, where u is either a node in T_D or empty node λ, similarly v is either a node in T_R or λ. Note that u and v cannot be an empty node at the same time, $u \rightarrow \lambda$ denotes deletion, and $\lambda \rightarrow v$ denotes insertion. Denote $D(F_1, F_2)$ as the edit distance between the subforest F_1 in T_D and the subforest F_2 in T_R. Let $T(r_1)$ and $T(r_2)$ be their rightmost subtrees rooted at nodes r_1 and r_2 respectively and let Φ be the empty forest, the dynamic programming process is:

- *Initialization:*

$$\begin{aligned}
&D(\Phi,\Phi)=0,\\
&D(F_1,\Phi)=D(F_1-r_1,\Phi)+\gamma(r_1\rightarrow\lambda),\\
&D(\Phi,F_2)=D(\Phi,F_2-r_2)+\gamma(\lambda\rightarrow r_2).
\end{aligned}$$

- *Iteration*:

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

3.1.2 Tree Node Mappings

We assign logical labels of nodes in the reference tree T_R to the nodes in the data tree T_D by tree node mapping. Mapping is the graphical specification of the edit operation applied to each node of the data tree. The optimal mappings M correspond to the sequence of edit operations that gives minimum edit distance $D(T_D,T_R)$. Without confusion, for any pair $(u\rightarrow v)\in M$, we denote $M(u)=v$. However we are only interested in the mappings related to the leaf nodes in T_D, each of which corresponds to a single zone in a document page image. For each node in the data tree T_D there are two possible operations: deletion or relabeling. Since we aim to find an optimal mapping from one leaf node in T_D to another leaf node in T_R, the deletion should be replaced by an appropriate relabeling, i.e., for each $(u\rightarrow\lambda)$ in M, we should replace it by $u\rightarrow v_o$, where v_o is the target node in T_R. And it is natural to map the deleted leaf node u to an appropriate undeleted leaf node u' in T_D and replace $(u\rightarrow\lambda)$ by $(u\rightarrow M(u'))$. We use a two step procedure to replace the deletion mappings.

- *Step 1:*

```
for each $(u \rightarrow v) \in M$
    if $l(u) = u$ and $v = \lambda$, then
        $v = M(u')$, such that
        $u' = arg \min_{w \in T_D, l(w)=w} \{\gamma(u \rightarrow w)\}$.
    endif
endfor
```

- *Step 2:*

```
for each $(u \rightarrow v) \in M$
    if $l(u) = u$, then
        $v = arg \min_{w \in des(v)} \{\gamma(u \rightarrow w)\}$.
    endif
endfor
```

Where $l(w)$ denotes the leftmost descendant of node w and $des(w)$ denotes the descendant leaf nodes of node w. Thus we have a leaf node to leaf node mapping for each leaf node in T_D but not the reverse.

3.1.3 Edit Cost Functions

The range of possible mappings M is constrained by the sibling and ancestor order in the two trees. On the other hand, the cost to delete, insert or relabel a tree node should depend on the structural and local features associated with each tree node. Among them the cost of deletion should be symmetric to the cost of insertion, the cost of relabeling should be symmetric about its two paramters. To be specific, we define:

$$\gamma(u\rightarrow v)=\gamma(v\rightarrow u)=\sqrt{\sum_{i=1}^{10}\frac{(f_i^u-f_i^v)^2}{s_i^2}}.\qquad(1)$$

So the edit cost to relabel u to v is equal to the Euclidean distance between normalized feature vectors $\mathbf{f}^u$ and $\mathbf{f}^v$. Let s_i^2 be the sample variance of i^{th} element of feature vector. We normalize the feature vector by its sample variance to eliminate the unbalanced value ranges of different feature elements.

$$\gamma(u\rightarrow\lambda)=\gamma(\lambda\rightarrow u)=\begin{cases}\sqrt{\sum\limits_{i=1,i\neq7}^{10}\frac{(f_i^u-f_i^{u'})^2}{s_i^2}}, \text{if} f_3^u=0,\\ \sqrt{\sum\limits_{i=1,i\neq6}^{10}\frac{(f_i^u-f_i^{u'})^2}{s_i^2}}, \text{if} f_3^u=1.\end{cases}\qquad(2)$$

Where u' is the left sibling of u if it exists, or null feature vector otherwise. We use the f_6 or f_7 to measure the spatial distance between u and u' according to the projection direction of their level. Binary feature f_3 indicates the projection direction. The definition of Equation 2 is consistent with Equation 1. Additionally we measure the cost to delete or insert a node according to its edit distance to its sibling node so that a locally unique node may survive the matching process.

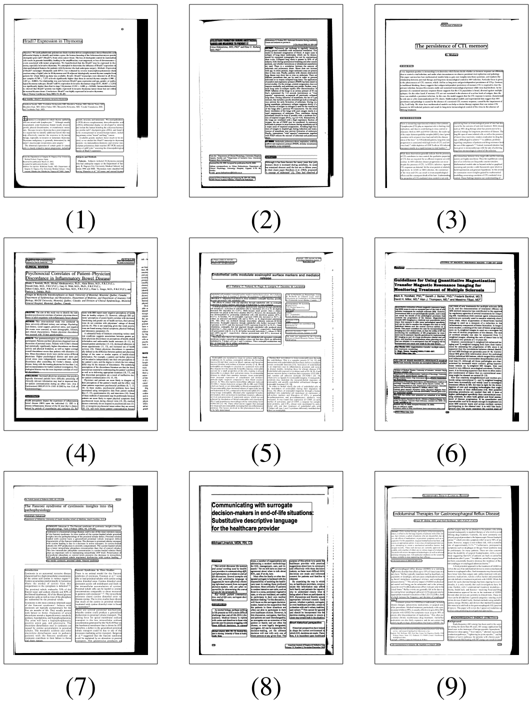
3.2. Layout Style Classification by Unsupervised Clustering

We use the K-medoids [5] algorithm to automatically cluster the training samples into distinct layout styles. A set of tree edit cost functions are defined based on Karl Person distance between two multivariate feature observations in [9]. This method is an unsupervised method and involves no training or manual selection of algorithm parameters. In this paper, we use more features in tree edit cost functions to better discriminate document pages of distinct styles.

4. Experiments

4.1. Preprocessing

We collected the first pages of 138 individual articles. These articles are from 9 different journals that have 9 distinct layout styles (as shown in Figure 2). In the preprocessing phase, a combinatorial OCR and zone segmentation module [2] was used to generate zones for each page. Local OCR features associated with each zone were extracted. We generate groundtruth for our dataset by manually assigning each zone with a logical label. After zone segmentation we built an ordered labeled X-Y tree for each document page and separated the trees into 3 parts: training set, cross validation set and test set that contain 51, 18 and 69 trees respectively. Table 1 shows the sizes of different datasets in our experiment.



(1) (2) (3)

(4) (5) (6)

(7) (8) (9)

Figure 2. Nine distinct layout styles.

4.2. Training Phase

There are two main steps in the training phase: (1) unsupervised clustering the training data set into K distinct layout styles; (2) a cross validation procedure to simultaneously choose the optimal size of compressed training set, which we denote as Q, and the number of nearest neighbor trees to a test tree (to be used in the recognition phase), which we denote as R.

First of all, we applied the K-medoids algorithm to the training set so that we obtained $K = 9$ clusters of trees [9]. Secondly, we compressed each cluster by choosing the Q nearest trees to the corresponding centroid tree and eliminating the others. We did this to prevent erroneously clustered trees. By global searching we simultaneously chose the optimal Q and R such that the proposed algorithm achieves the highest accuracy to the cross validation set. The optimal value of Q and R obtained for the training set are 5 and 1, respectively.

Table 1. Number of document pages in training, cross validation, and test datasets of each layout style.

Layout Style	1	2	3	4	5	6	7	8	9
Training	4	5	4	9	8	6	7	3	5
Cross Validation	1	2	1	3	3	2	3	1	2
Test	4	8	4	12	11	8	10	5	7

4.3. Recognition Phase

In the recognition phase, each test tree is classified into the layout cluster whose centroid tree has a minimum edit distance to the test tree. Consequently, we match the test tree to the Q trees found in the training phase for this cluster, and from the Q trees we choose the R trees that are nearest to the test tree. Then the selected R trees cast votes on the logical labels of leaf nodes in the test tree according to the obtained tree node mappings as described in Section 3.1.2. Each leaf node is labeled as the logical entity that has the maximum number of votes. Let us denote $L(u)$ as the logical label of zone u, $M_r(u)$ as the corresponding mapped zone in the r^{th} tree, and the designed code book of logical label as Σ. The process to obtain the logical label of zone u in the test tree can be expressed as: $L(u) = arg \underset{j \in \Sigma}{max} \left\| \{ r | L\left(M_r(u)\right) = j, 1 \leq r \leq R \} \right\|$.

4.4. Results

The results of experiment show that the overall accuracy of logical entity recognition is 93.40% over the 1,015 test zones. And the recognition accuracies of individual types of logical entity are evaluated as well. See Table 2.

Table 2. Recognition Accuracies for Different Logical Entities

Title	Authors	Affiliations	Abstract
94.81%	92.96%	95.96%	98.82%
Section	Running Header	Footnote	Page Number
86.56%	98.05%	88.37%	95.45%

In addition, we set up a balanced training data set of 35 pages. The small balanced training set consists of 4 pages of style 1,2,3,4,5,6,7,9 respectively and 3 pages of style 8. All of them were randomly chosen from the training set in Table 1. The recognition rate using the small balanced training set is 92.31%.

We compare the performance of our algorithm with that of Support Vector Machines (SVMs) without considering layout style information. We combine the training set and cross validation set in Table 1 into one training set and use the local font features of the zones to train the support vector classifier. We chose the radial basis function as the kernel function and selected the penalty parameter C and the kernel coefficient γ by a 5-fold cross validation in the training set. The optimal C and γ are 32 and 8 respectively. The trained classifier is used on the test set in Table 1 and average logical entity recognition accuracy is only 79.31%. See [3] for the details of this algorithm.

4.5. Error Analysis

Error distributions in individual logical entity types are listed in Table 3. The cell element (i, j) of this table records the ratio of errors when logical entity i was recognized as logical entity j over all the errors of logical entity i. From the table we can see that the segmented zones are prone to be misclassified as the categories that are close to it physically, such as "title" to "authors", "authors" to "affiliations", "affiliations" to "abstract", which is consistent with tree matching algorithm. We also see that most errors are due to three factors: (1) zone segmentation erroneously merges different logical entities or missed part of a logical entity; (2) horizontal gaps between paragraphs in different columns are accidentally aligned horizontally and are considered as a single horizontal gap, which results in an incorrect tree being built; and (3) floating logical entities, such as footnote and page number, are located at different sides in odd and even pages.

Table 3. Error analysis

Logical label	1	2	3	4	5	6	7	8
1	/	100%	0	0	0	0	0	0
2	20%	/	80%	0	0	0	0	0
3	0	10%	/	50%	20%	0	20%	0
4	0	0	60%	/	40%	0	0	0
5	0	2%	21%	72%	/	0	5%	0
6	100%	0	0	0	0	/	0	0
7	0	0	17%	0	50%	0	/	33%
8	0	0	0	0	0	0	100%	/

1:*Title*, 2:*Authors*, 3:*Affiliations*, 4:*Abstract*,
5:*Section*, 6:*Running Header*, 7:*Footnote*, 8:*Page number*.

5. Conclusion and Future Work

In this paper we have presented a novel logical entity recognition algorithm based on ordered labeled X-Y tree matching and unsupervised layout style recognition. The overall accuracy is high as 93.40% when using the unbalanced training set and 92.31% when using the balanced but smaller training set. These figures are much higher than the accuracy of SVM using the same data set. The experimental result proved the efficiency of selected local font features and the robustness of X-Y tree representation. The layout style and logical entities are recognized simultaneously with satisfactory accuracies.

Apparently there is still space to enhance this algorithm. Firstly erroneously clustered layout styles can be reduced if X-Y tree representation is adjusted to avoid the three clusters of errors discussed in Section 4.5. Secondly parsing techniques can be combined to improve the logical entity recognition. In our future work we will generalize the algorithm to more categories of logical labels and evaluate it on general article document collection like UW Document Image Database III.

Acknowledgement
This research was supported by the Intramural Research Program of the National Library of Medicine, National Institutes of Health.

References

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

[2] S. E. Hauser, D. X. Le, and G. R. Thoma. Automated zone correction in bitmapped document images. In *The International Society for Optical Engineering Document Recognition*, pages 248–258, San Jose, CA, January 2000.

[3] C. Hsu, C. Chang, and C. Lin. A practical guide to support vector classification.

[4] Y. Ishitani. Logical structure analysis of document images based on emergent computation. In *Proceedings of International Conference on Document Analysis and Recognition*, pages 189–192, Bangalore, India, September 1999.

[5] L. Kaufman and P. Rousseeuw. *Finding groups in data: An introduction to cluster analysis*. Wiley, New York City, New York, 1990.

[6] J. Kim, D. X. Le, and G. R. Thoma. Automated labeling in document images. In *Proceedings of SPIE Conference on Document Recognition and Retrieval VIII*, pages 111–122, San Jose, CA, January 2001.

[7] 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*, 15:737–747, 1993.

[8] J. Liang, D. Doermann, M. Ma, and J. Guo. Page classification through logical labeling. In *Proceedings of International Conference on Pattern Recognition*, pages 477–480, Quebec, Canada, August 2002.

[9] S. Mao, L. Nie, and G. R. Thoma. Unsupervised style classification of document page images. pages 510–513, Genova, Italy, September 2005.

[10] D. Niyogi and S. N. Srihari. Knowledge-based derivation of document logical structure. In *Proceedings of International Conference on Document Analysis and Recognition*, pages 472–475, Montreal, Canada, August 1995.

[11] K. Summers. Near-wordless document structure classification. In *Proceedings of International Conference on Document Analysis and Recognition*, pages 462–465, Montreal, Canada, August 1995.

[12] Y. Tateisi and N. Itoh. Using stochastic syntactic analysis for extracting a logical structure from a document image. In *Proceedings of International Conference on Pattern Recognition*, pages 391–394, Jerusalem, Israel, October 1994.

[13] K. Zhang, D. Shasha, and J. T. L. Wang. Approximate tree matching in the presence of variable length don't cares. *Journal of Algorithms*, 26:33–66, 1994.