Logical Labeling of Document Images Using Layout Graph Matching with Adaptive Learning

Jian Liang and David Doermann

Institute for Advanced Computer Studies
University of Maryland at College Park
{lj, doermann}@cfar.umd.edu

Abstract. Logical structure analysis of document images is an important problem in document image understanding. In this paper, we propose a graph matching approach to label logical components on a document page. Our system is able to learn a model for a document class, use this model to label document images through graph matching, and adaptively improve the model with error feed back. We tested our method on journal/proceeding article title pages. The experimental results show promising accuracy, and confirm the ability of adaptive learning.

1 Introduction

A typical document analysis system consists of page segmentation, optical character recognition (OCR), and logical structure understanding. The interest in the logical structure has been inspired by the emergence and popularity of common representation standards such as XML. With such standards, it is possible to encode complicated structural information together with content. The logical structure extracted from existing documents and expressed in XML format will greatly enhance the way the content can be used. In addition to traditional full text search, structured search and browsing, routing, database input, data mining, and document reformatting will be possible. Most current document analysis systems, however, cannot achieve this goal. Their output is typically no more than an unstructured text stream, although some systems try to make the output document layout similar to the original page by controlling font sizes, font styles, and text block positions. We need new tools to understand the logical structure of document images.

The task of logical labeling is to label segmented blocks on a document image as title, author, header, text column, etc. The set of labels will depend on document classes and/or applications. Logical labeling techniques can be roughly characterized as either zone-based or structure-based. Zone-based techniques [1][8] classify zones individually based on features of each zone. Structure-based techniques incorporate global constraints such as position. These techniques can further be classified as either top-down decision based [3][7], or bottom-up inference-based techniques [2][5]. Global optimization techniques [6][9] are often hybrids of the first two.

Based on our observation that page layouts tend to be consistent within a document class, we create a layout graph model for each visually distinct document class. Logical labeling is accomplished through a matching from a candidate page to a model.

D. Lopresti, J. Hu, and R. Kashi (Eds.): DAS 2002, LNCS 2423, pp. 224–235, 2002.

2 System Overview

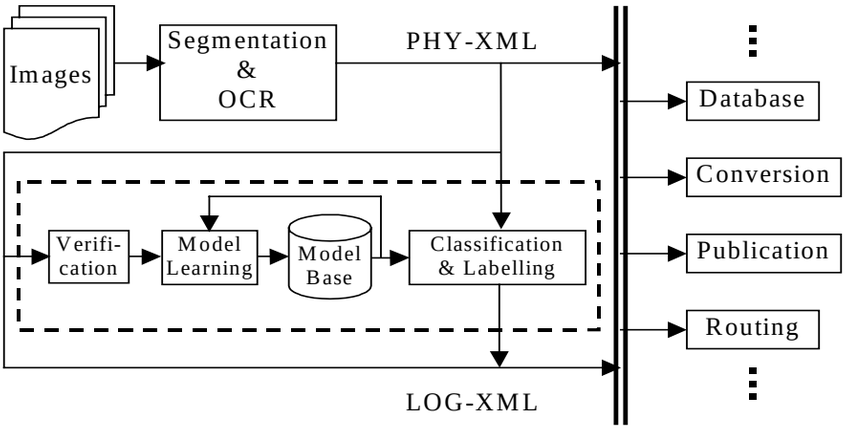


Fig. 1. System overview

Fig. 1 shows an overview of our document analysis system. First, document images are processed by a segmentation-and-OCR engine. We assume that the results are reasonably good. In particular, mild over-segmentation is acceptable, while under-segmentation which crosses logical content is not welcome. The outcome XML file (PHY-XML in the figure) contains information about the physical layout and text content of the original document page. The LOG-XML in the figure stands for logical structure XML file, which contains information about document class and logical labels corresponding to the PHY-XML file.

Our focus is on the logical labeling modules inside the dashed line frame. There are three working processes:

1. *Model Initialization*: At the beginning, there is no model in the model base. The user needs to provide ground truth LOG-XML files corresponding to the PHY-XML files, which are taken by the model learning module to learn an initial document model. Document models are represented by layout graphs (see section 3).
2. *Logical Labeling and Adaptive Model Learning*: Once there is a model in the model base, a new document image (represented by a PHY-XML file after segmentation and OCR) can be first converted into a candidate layout graph, then matched to the model in order to assign logical labels to each block, resulting in a LOG-XML file. The PHY-XML and LOG-XML files can be used by downstream applications. If there is any error, the user verifies the PHY-XML and LOG-XML files. The verification results, along with the model, are handed over to the model learning module to improve the model.
3. *Classification and Labeling*: More than one document model is stored in model base, each representing a document class. An incoming document (represented by a PHY-XML file after segmentation and OCR) is converted to a candidate layout graph and matched to all models. The best match gives both the labeling and the classification result. The verification and adaptive model learning remain the same.

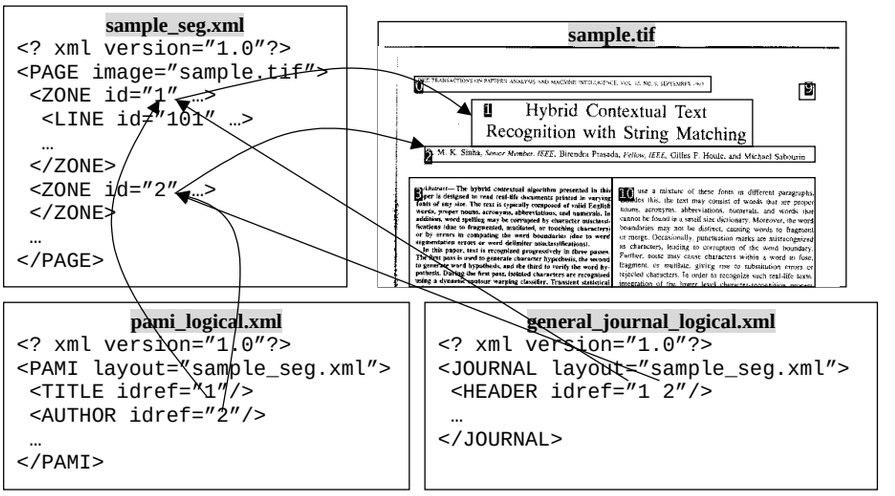


Fig. 2. XML representation of segmentation-OCR results and logical structure

Fig. 2 shows fragments of an example document page (*sample.tif*), the corresponding PHY-XML file (*sample_seg.xml*), and two LOG-XML files (*pami_logical.xml* and *general_journal_logical.xml*).

The PHY-XML file has a hierarchical structure, starting with a PAGE, down to ZONEs, LINEs, WORDs, and ending at CHARs. Each element has attributes describing its bounding box and average font size. The CHAR element has a character code attribute.

Each PHY-XML file can have more than one logical view. In *pami_logical.xml*, the TITLE and the AUTHOR are distinguished; but in *general_journal_logical.xml*, it is enough to identify the HEADER. Multiple logical views could be obtained through simple conversion: the combining of TITLE and AUTHOR into HEADER; or using multiple models: a PAMI model, and a GENERAL_JOURNAL model. Since the PAMI model has to achieve a finer resolution than the general one, it is more likely to produce errors. Therefore, if a general logical structure is the final goal, it would be better to use a general model.

As a simple sample application, we use XML stylesheets to translate the PHY-XML and LOG-XML files into other formats, such as HTML. Different tags can be assigned for text with different logical labels such that they are displayed with different font size/style and alignment. Depending on the application, different stylesheets can be designed. As an example, Fig. 3 shows one original document page and the converted HTML result.

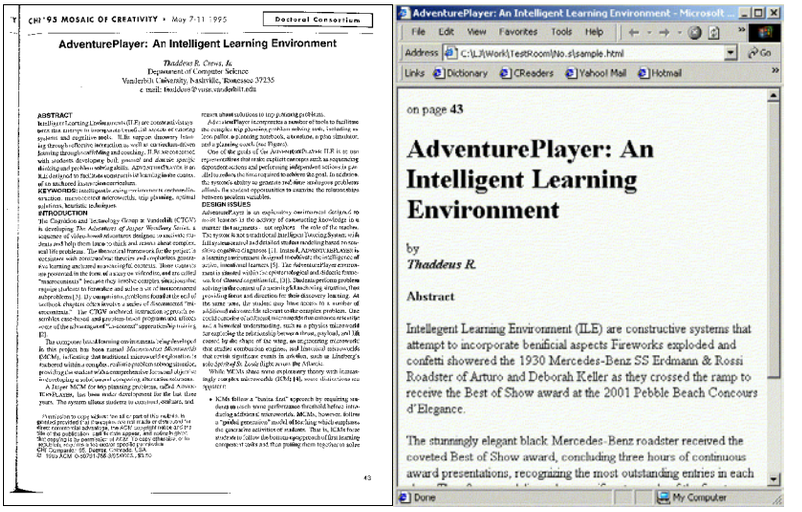


Fig. 3. Original document page and converted HTML result

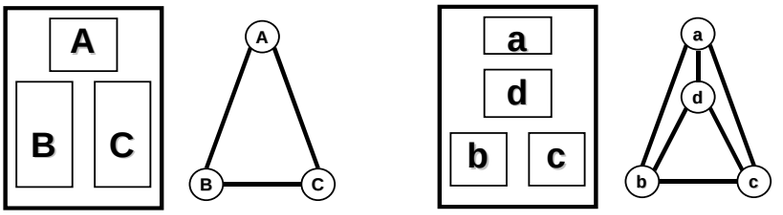


Fig. 4. Example layout and layout graphs

3 Layout Graph Matching

3.1 Layout Graph

A layout graph is a fully connected attributed relational graph. Each *node* corresponds to a segmented block on a page. The attributes of a node are the position and size of the bounding box, and the normalized font size (*small*, *middle*, or *large* as compared to the average font size over the whole page). An *edge* between a pair of nodes reflects the spatial relationship between two corresponding blocks in the image. We decompose this spatial relation into relations between block edges. For example, the relation between two vertical edges is normalized as *left-of*, *aligned*, or *right-of*. Similarly, the relation between two horizontal edges is *above*, *aligned*, or *below*. With 9 normalized edge relations (4 for left/right edges, 4 for top/bottom edges, and 1 for vertical central lines) we can describe the spatial relation between two blocks. If the distance between two block edges is within a threshold (we used 20 pixels for 300DPI images), they are regarded as *aligned*. The threshold depends on the skew of the image, and the document class.

Fig. 4 shows two layout graphs (attributes not shown). The attributes of edge *AB* in the left graph are shown in Fig. 5.

A model layout graph has some extra features. Each node carries a logical label, such as TITLE, AUTHOR, or ABSTRACT. There is a weight factor associated with each node/edge attribute. There is a *null*-cost defined for each node if it is not mapped to any other node in a candidate layout graph in a match.

Edge of block A	*Relationship*	*Edge of block B*
Left	To-the-right-of	Left
Left	To-the-left-of	Right
Right	To-the-right-of	Right
Right	To-the-left-of	Left
Top	Above	Top
Top	Above	Bottom
Bottom	Above	Bottom
Bottom	Above	Top
Vertical centre	To-the-left-of	Vertical centre

Fig. 5. Example edge attributes

3.2 The Match Cost

Typically, we have more nodes in a candidate layout graph than in the model graph. Thus we allow several nodes in a candidate graph be mapped to one node in a model graph. For example, in Fig. 4, let *M(A,B,C)* be the model graph, and *U(a,b,c,d)* be the candidate graph. Each of *a*, *b*, *c*, and *d* can be mapped to either *A*, *B*, *C*, or *null*. There are $4^4=256$ possible mappings. Here are two examples:

(A-a, B-b, C-c, A-d), (A-Φ, B-d, C-a, Φ-b, Φ-c)

We need a metric to measure which mapping is the best. For a given mapping, an intermediate layout graph, *T*, is first constructed based on *U* such that the mapping between *T* and *M* is 1-1. Then a cost is computed for the 1-1 mapping and defined as the quality measurement of the mapping between *U* and *M*. The best match is the one with minimal cost.

Let $U = \{u_i\}_{i=1}^{K}$ be the candidate layout graph, $M = \{m_i\}_{i=1}^{L}$ be the model layout graph. The N-1 mapping from *U* to *M* is $f: U \to M$. The inverse mapping is defined as $f^{-1}(m) = \{u \mid u \in U, f(u) = m\}$. For a given subset of layout graph nodes, *S*, define a grouping operation *g* such that *g(S)* represents a grouped node. In particular, the bounding box of *g(S)* is the union of bounding boxes of nodes in *S*, and the font size of *g(S)* is the average of font sizes of nodes in *S* weighed by character numbers of each node. If *S* is a null set, *g(S)* is simply null, too. Then, the intermediate layout graph, *T*, is defined as $T = \bigcup_{m \in M} g(f^{-1}(m))$, where $g(f^{-1}(m))$ is the grouped node corresponding to *m*. Once nodes of *T* are determined, the edge attributes (i.e., spatial relationship between nodes in *T*) are computed. After that, *T* is fully determined.

Now the mapping from *T* to *M* is a 1-1 mapping, $h: T \to M$, deduced from $f: U \to M$. The match cost is defined as follows:

For a pair of mapped nodes, the cost is defined as the sum of differences between

corresponding attributes, weighted by the weight factors in model node. If a model node is mapped to Φ, then a *null-cost* is incurred.

A cost is similarly defined for a pair of edges. A zero cost is defined if an edge is mapped to Φ.

The graph match cost is the sum of all node pair costs and edge pair costs. That is

$$C(T,M,h) = \sum_{t \in T} C_{t,h(t)} + \sum_{t \in T} \sum_{s \in T} C_{ts,h(t)h(s)} \quad , \tag{1}$$

where $C_{t,h(t)}$ is the node pair cost,, and $C_{ts,h(t)h(s)}$ is the edge pair cost.

Once the best match is found, the logical label associated with each node m in the model is assigned to every node of $g(f^{-1}(m))$ in the candidate layout graph. If we have more than one model, each will give its best match and cost for a given U.

Given one model and one candidate, logical labeling is simply the search for the best match. Given multiple models, page classification is equivalent to the selection of the model with lowest match cost. The key is to find proper attributes and weights.

4 Finding the Match

Graph matching in general is NP-hard, and is itself an interesting problem. Practical solutions either employ branch and bound search with the help of heuristics, or non-linear optimization techniques [4]. It is even more difficult to do N-1 matching as required in our approach.

Our approach is a two-step approximate solution that aims at sub-optimal N-1 match. First, we search for the best 1-1 match from U to M, where U is the candidate graph, and M is the model graph. There are usually some unmatched nodes left in the candidate graph. In the second step, they are grouped to the matched nodes to form the intermediate graph, T. The match between T and M is a 1-1 mapping, determined by the best 1-1 mapping from U to M. Different grouping schemes result in different T. We search for the scheme that results in the best 1-1 match from T to M. From this grouping we can go back to get the sub-optimal N-1 match from U to M. Although in general it is sub-optimal, in our experiments it is usually satisfactory.

To address the computational expense involved in the first step, we took advantage of the fact that 1-1 match cost can be computed in an incremental way, and designed an efficient branch-and-bound search strategy. In experiments, the search took usually less than one second.

In the second step, we dramatically decreased the number of possible grouping schemes using the constraint that nodes in T should not overlap. The search in second step usually takes tens of milliseconds. Our assumption is that different logical parts on a page do not overlap, which is satisfied in most Manhattan style publications such as journals, proceedings, and business documents. Since this constraint is associated with each edge in the model graph, it is possible to relax it for some edges while keeping it for the rest. In our experiments, however, we imposed the non-overlapping constraint universally.

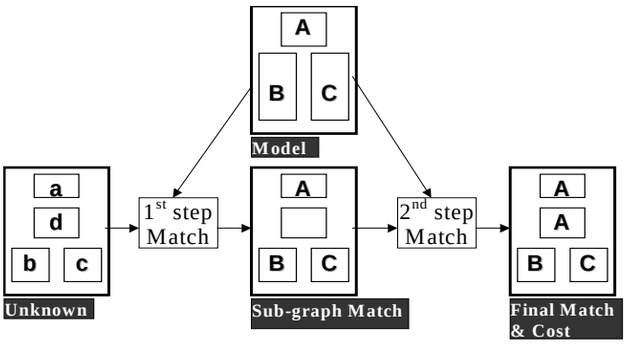


Fig. 6. Two-step N-1 graph matching procedure

5 Adaptive Model Learning

The model layout graph plays a key role in our matching procedure. If a model is ill defined, not only the result will be bad, but also the search for the best match may take a long time. It is important to find the appropriate features and weight factors. In our design, it is not too hard to write and/or adjust a model completely manually. However it is more time/cost efficient if accomplished automatically. In our previous work [10] we focused on the initialization of a model from a set of ground truth samples. Recently we continued to study adaptive model learning. For example, if the mistakenly labeled "author" is above "title", we could increase the weight associated with their vertical spatial relationship to force "author" be placed below "title". Inspired by the error back propagation learning methods in training neural networks, we designed an adaptive learning algorithm for improving models.

Given a model, a sample segmentation, and the ground truth mapping between them, we start out finding the N-1 match between the segmentation and the model using the methods we presented in the section above. We then compare the result with the ground truth. If they are different, it means the false match somehow has a lower cost than the true match. Then we need to modify the model. The step size of the modification is small to avoid abrupt jumps.

To summarize, our model updating policy is the following (see Fig. 7):

Let G be the feature in ground truth, R be the feature in match result, M be the feature in model, and W be the weight factor.
If $G=R$, leave M and W intact.
Else, bias M towards G by a small preset step.
And if $|M-G| < |M-R|$, increase W by a small preset amount.
Or, i.e., $|M-G| > |M-R|$, decrease W by a small preset amount.

With the adaptive learning ability, it is possible to initialize a model with only one sample image, then to improve the model as new samples come in one at a time. If any errors occur, there could be different strategy as to how much training should be done. The sample could be used only once to improve the model, it could be trained upon until no error (unnecessarily the first error) occurs, or it could be trained upon for a certain times. Alternatively a number of the most recent seen samples are all passed into the training procedure to avoid over fitting to the new sample.

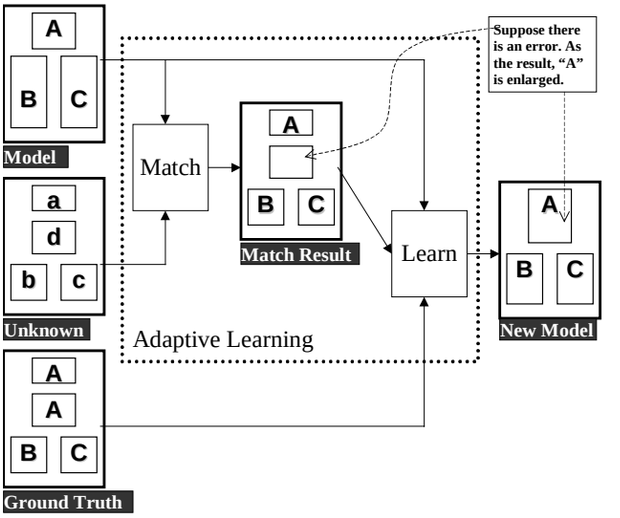


Fig. 7. Adaptive model learning procedure

6 Experiments

Our data set in this experiment consists of 80 title pages from 4 journal/conference proceedings, 20 pages from each. All of them are black-and-white images scanned at 300 DPI. In Fig. 8 four representative pages are shown. They share a common layout of general two-column journal title pages, yet slight differences exist in running headers, footers, page numbers, etc. Segmentation is done by off the shelf commercial OCR product from ScanSoft. We only used the position of segmented blocks, not OCR result or font information.

We adopted the leave-one-out strategy to achieve near real life testing environment. That is, for each one of the 20 pages, we pick it out as testing sample for once and use the other 19 as training samples. In this way we can get 20 testing results and then average them out. More precisely, suppose

1. $S = \{P_j; j=1,\ldots,\|S\|\}$ is the set of all sample pages, where P_j is the jth page plus ground truth labels.
2. For each $s=1,\ldots,\|S\|$, S is divided into two subsets: $TE_s = \{P_s\}$, and $TR_s = S - TE_s$.
3. The ith generation model for testing sample Ps is learned as
   if i=1 $m_s^{(i)} = \text{Initialize}(TR_s)$,
   otherwise $m_s^{(i)} = \text{Improve}(TR_s, m_s^{(i-1)})$
4. The i^{th} generation testing result for sample P_s is $R_s^{(i)} = \text{Match}(P_s, m_s^{(i)})$.
5. The i^{th} generation error count is $e_s^{(i)} = \text{Compare}(R_s^{(i)}, P_s)$.
6. The number of blocks in P_s is b_s.
7. The i^{th} generation average error rate is defined as $E_s^{(i)} = \sum_{s=1}^{\|S\|} e_s^{(i)} \Big/ \sum_{s=1}^{\|S\|} b_s$.

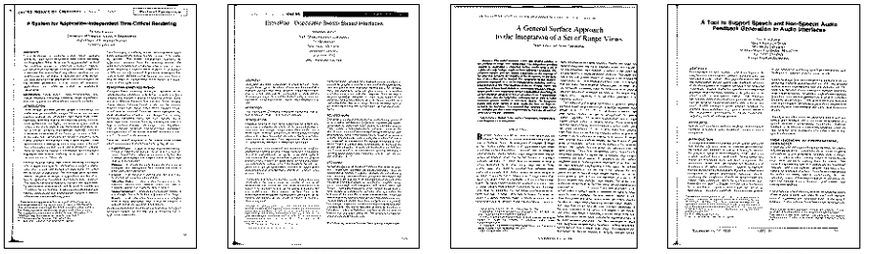
Fig. 10 (a) shows one sample page; (b) shows a visualization of the model; (c) is the labeling result after first step sub-graph matching; and (d) is the final result after

second step matching. Another set of results is illustrated in Fig. 11. In Fig. 9 the relation between average labeling error rates and the numbers of training cycles is shown. After 10 cycles error rates dropped 30% on average, which confirms the effectiveness of our adaptive learning algorithm.

7 Conclusion

We have presented a model driven approach for logical labeling of document images, focusing on the adaptive model learning ability. We have demonstrated promising results using only simple attributes like bounding box position, size, and font size. This suggests that global optimization is a powerful tool in dealing with noisy and variant data. It could be expected that when more features are utilized, such as font style, text content, better result would be achieved.

One of our ongoing tasks is to investigate the match cost and use it as a distance measure to classify/reject documents. Another one is to build hierarchical model base whose leaves are specific models (as PAMI) and non-terminal nodes are unified models (as general journal). Page classification and logical labeling can then be done in an integrated, and hierarchical way.



CHI'95 CHI'96 PAMI UIST'95

Fig. 8. Sample pages

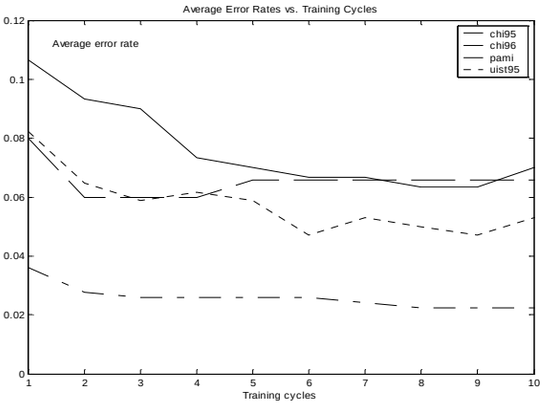
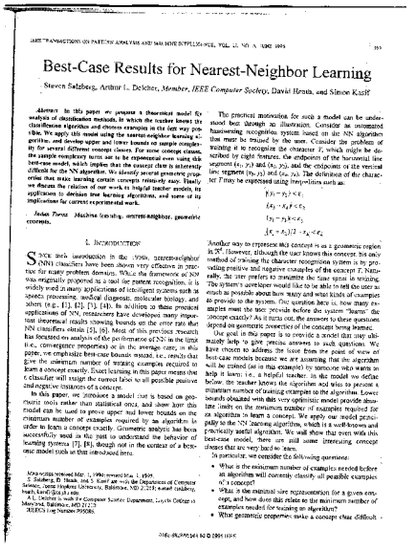
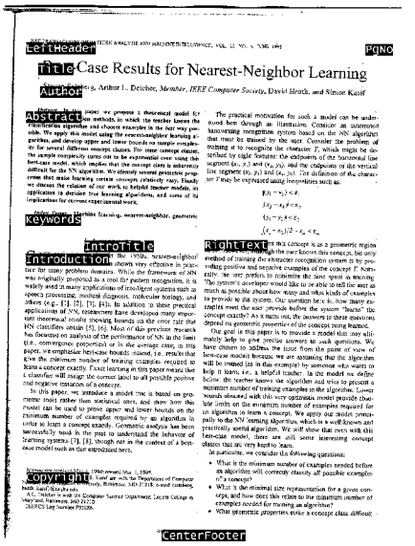


Fig. 9. Average error rates vs. training cycles

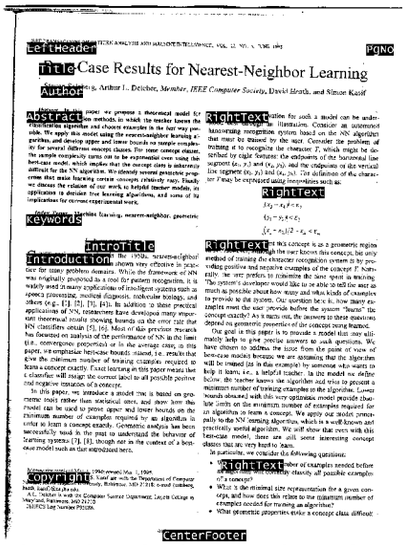


(a) Original image with segmentation box overlaid

(b) Visualization of model



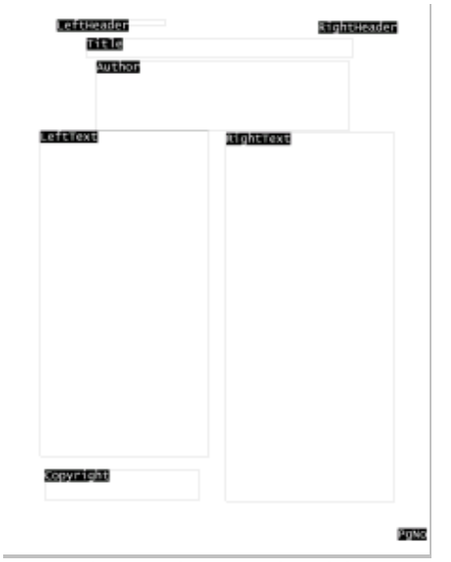
(c) Result of sub-graph matching



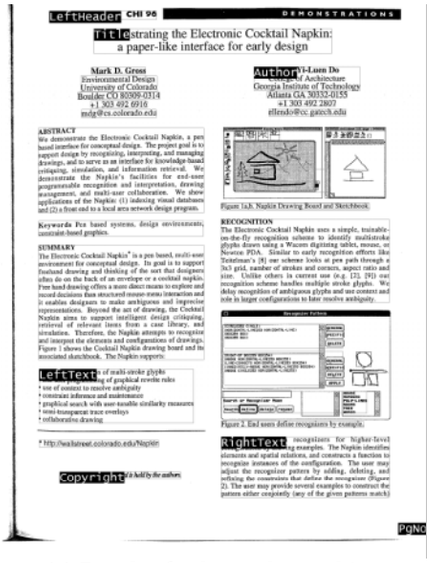
(d) Final result of N-1 graph matching

Fig. 10. Example image, model, and labeling result (I)

(a) Original image with segmentation box overlaid



(b) Visualization of model



(c) Result of sub-graph matching

(d) Final result of N-1 graph matching

Fig. 11. Example image, model, and labeling result (II)

References

1. O. Altamura, F. Esposito, and D. Malerba. “Transforming paper documents into xml format with WISDOM++”. Journal of Document Analysis and Recognition, 2000, 3(2):175–198.
2. T. A. Bayer and H. Walischewski. “Experiments on extracting structural information from paper documents using syntactic pattern analysis”. In Proceedings of The Third International Conference on Document Analysis And Recognition, 1995, pp. 476–479.
3. A. Dengel, R. Bleisinger, F. Fein, R. Hoch, F. Hones, and M. Malburg. “OfficeMAID – a system for office mail analysis, interpretation and delivery”. In International Workshop on Document Analysis Systems, 1994, pp. 253–276.
4. S. Gold and A. Rangarajan. “A graduated assignment algorithm for graph matching”. IEEE Trans. Pattern Anal. Machine Intell., 1996, 18(4):377–388.
5. T. Hu and R. Ingold. “A mixed approach toward an efficient logical structure recognition from document images”. Electronic Publishing, 1993, 6(4):457–468.
6. Y. Ishitani. “Model-based information extraction method tolerant of OCR errors for document images”. In Proceedings of The Sixth International Conference on Document Analysis And Recognition, 2001, pp. 908–915.
7. M. Krishnamoorthy, G. Nagy, S. Seth, and M. Viswananthan. “Syntactic segmentation and labeling of digitized pp. from technical journals”. IEEE Transactions On Pattern Analysis And Machine Intelligence, 1993, 15(7):737–747.
8. G. I. Palermo and Y. A. Dimitriadis. “Structured document labeling and rule extraction using a new recurrent fuzzy-neural system”. In Proceedings of The Fifth International Conference on Document Analysis And Recognition, 1999, pp. 181–184.
9. H. Walischewske. “Learning regions of interest in postal automation”. In Proceedings of The Fifth International Conference on Document Analysis And Recognition, 1999, pp. 317–340.
10. J. Liang and D. Doermann. “Page classification through logical labeling”. (To be published) In Proceedings of The International Conference of Pattern Recognition, 2002.