A Statistical Learning Approach To Document Image Analysis

Kevin Laven
Dept of Computer Science
University of Toronto
klaven@cs.toronto.edu

Scott Leishman
Dept of Computer Science
University of Toronto
scott.leishman@utoronto.ca

Sam Roweis
Dept of Computer Science
University of Toronto
roweis@cs.toronto.edu

Abstract

In the field of computer analysis of document images, the problems of physical and logical layout analysis have been approached through a variety of heuristic, rule-based, and grammar-based techniques. In this paper we investigate the effectiveness of statistical pattern recognition algorithms for solving these two problems, and report results suggesting that these more complex and powerful techniques are worth pursuing. First, we developed a new software environment for manual page image segmentation and labeling, and used it to create a dataset containing 932 page images from academic journals. Next, a physical layout analysis algorithm based on a logistic regression classifier was developed, and found to outperform existing algorithms of comparable complexity. Finally, three statistical classifiers were applied to the logical layout analysis problem, also with encouraging results.

1. Background

Document image understanding is the process of automatically extracting useful information from page images. The problem can be broken down into a series of subproblems, with each working from the results of the previous. Two key steps in such a series are *segmentation* (or page physical structure analysis), in which regions of ink are identified, and *labeling* (or page logical structure analysis), in which these regions are assigned meaningful labels.

Many existing algorithms for segmentation and labeling involve heuristic or rule-based approaches[4], sometimes enhanced by decision trees[1] or page grammars[2]. While these approaches have met with some success, they do not always generalize well to documents outside the development set. In addition, the complexity of these techniques makes them difficult to replicate, making quantitative performance comparisons nearly impossible.

In response to these difficulties, Song Mao et al. have called for an approach to these problems based on formal models[4]. They note specific advantages including the ability to estimate parameter values for the model from training data, and the possibility of selecting a model of appropriate complexity for the class of documents being examined. This paper describes approaches to both segmentation and labeling based on formal models for statistical pattern recognition.

2. Ground Truth Data

We have produced a data set of ground truth segmentations and labelings of scanned pages from academic journals. In our data, each region of ink is labeled with one of 25 precise region labels, which can be aggregated into a smaller set of 4 generalized labels.

Our segmentations followed the "Manhattan" style of page layout, under which all regions are rectangular and aligned with the edges of the page. Nested or cyclical regions [1] are not permitted. This type of layout is common in academic journals (especially in natural sciences and engineering), which are the focus of our applications.

A new page tagging software package called JTAG has been developed for the purpose of creating such ground truth data by hand-labeling.

2.1. The JTAG Software Package

The JTAG (Journal TAGging) software system shown in Figure 1 is a collection of TCL scripts which allow for the creation of rectangular page-aligned regions, and for the assignment of logical labels to each region.

A new file format was developed for storing and exchanging the tagging information. Each .jtag file is a text file corresponding to a single page image. The file contains a record of each ink region in the file, including the four boundaries of the region, the label assigned, and some additional parameters used by the JTAG software. If any preprocessing algorithms are applied to the page, their settings can also be recorded in the file.

Automated segmentation and labeling algorithms are trained on the hand labeled data, and can be applied to segment and label future pages, including those destined

1 Cyclical regions are a group of regions arranged such that no line can be drawn separating two of the regions that does not intersect another region.

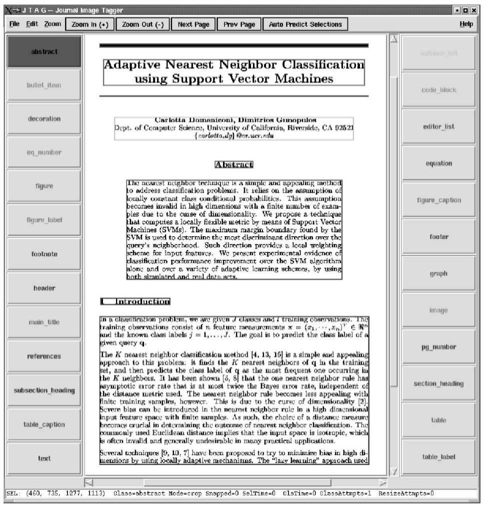

Figure 1. Screenshot of JTAG software.

for hand labeling. This "bootstrapping" reduces the amount of manual work required to produce ground truth data, as the automated algorithms can suggest a preliminary version of the segmentation and labeling, which need only be corrected by the human tagger. Tagging rates of about one page per minute can be achieved with this system.

2.2. Details of the Data Set

The data set consists of images of pages from articles appearing in the Journal of Machine Learning Research (JMLR)[2] and the proceedings of the Neural Information Processing Systems conference[3] in 2001 and 2002. Each of these two sources employs its own standard single-column format, which authors generally adhere to, although some articles deviate from the standards. The JMLR data set consists of 15 articles, with 472 hand-labeled page images containing 5,556 individual ink regions. The NIPS data set consists of 58 articles, with 460 hand-labeled page images containing 4,916 ink regions. The articles in each data set was divided into training data (about 2/3 of the articles) and test data (the remaining 1/3 of the articles).The logical region labels are taken from the set of 25 precise categories, which are aggregated into 4 generalized categories, as shown in Table 1. The precise labels were those we anticipated would be needed by applications such as targeted document image retrieval, whereas the generalized labels were those likely to be necessary for other applications, such as preprocessing for optical character recognition.

2 http://jmlr.csail.mit.edu/
3 http://www.nips.cc/

Precise Category	Generalized Category
text	*TEXT*
header	*TEXT*
section heading	*TEXT*
subsection heading	*TEXT*
figure label	*TEXT*
figure caption	*TEXT*
references	*TEXT*
abstract	*TEXT*
bullet item	*TEXT*
page number	*TEXT*
main title	*TEXT*
footer	*TEXT*
table label	*TEXT*
table caption	*TEXT*
editor list	*TEXT*
equation number	*TEXT*
author list	*TEXT*
footnote	*TEXT*
decoration	*OTHER*
equation	*EQUATION*
table	*FIGURE*
image	*FIGURE*
graph	*FIGURE*
figure	*FIGURE*
code block	*FIGURE*

Table 1. The 25 precise labels used, and the 4 generalized labels into which they were aggregated.

3. Segmentation

Most segmentation algorithms can be described as either "top-down" or "bottom-up". Top-down approaches begin by considering the entire page as one region, successively dividing regions into smaller ones until a termination criteria is met. Nagy's xycut algorithm[6] provides the basis for many such systems. Bottom-up approaches start by dividing the page into small components, such as individual marks of ink, and then merge these together to form regions. The Area Voronoi diagram technique[3] is an example of such a system.

Below, we propose a new *Concurrent Learn-To-Cut* algorithm that can be applied to any Manhattan layout page. We compare this new algorithm to the relatively simple but general algorithms that underlie most existing complex segmentation systems.

3.1. Benchmark Algorithms

We implemented three established segmentation algorithms to act as performance benchmarks: Nagy's xycut algorithm, the Area Voronoi Diagram technique, and a simple "smearing" algorithm. In smearing, ink is duplicated x pixels across the page, and y pixels up and down the page. Connected components after smearing are identified, and bounding boxes placed around them.

Each of these algorithms contains two free parameters which were optimized for each particular dataset. Given that these parameters are integers with a finite range of plausible values, it was possible to determine the best values for a given data set, based on simple "matching" performance metric (described below). This optimization was done for each algorithm, on each set of training data, allowing us to test the generalization of these parameters to a separate set of test data.

3.2. Concurrent Learn-To-Cut

The Concurrent Learn-To-Cut (LTC) algorithm treats page segmentation as a supervised learning problem, where the system learns to segment pages following the patterns laid out in a training set of hand-segmented pages. The LTC algorithm is a top-down algorithm restricted to producing Manhattan layouts. It employs a logistic regression classifier to take advantage of large amounts of information when deciding which cuts should be made.

Approaching page segmentation with a supervised learning algorithm presents two challenges. First, the segmentation problem must be reduced to a straightforward classification question. Second, some manner must be found of creating representative training data from the ground truth segmentations.

Top-down segmentation can be reduced to answering the question "should this cut be made", through the use of an appropriate structural algorithm. The algorithm we selected is based on the version of the xycut algorithm used in [2]. The horizontal projection profile is used to find rows of whitespace across the page, which are called *cut candidates*. A vector of numerical values (called a feature vector) describing each cut candidate is extracted, based on the dimensions and position of the cut, as well as the ink density in several predetermined rows and blocks of pixels near the cut. This feature vector is passed to a logistic regression classifier, which labels each cut as either *valid* or *invalid*. All of the cuts labeled *valid* are then made. This process is repeated on all of the new segments that result, alternating between making vertical cuts and horizontal cuts. The recursion continues on each new segment generated, alternating cut directions, until no cuts are selected as *valid* in a pass. One pass of the algorithm is illustrated by Figure 3.

Training data, in the form of correctly classified feature vectors, can be created from hand-segmented pages. An algorithm very similar to the one above is applied, except that cut candidates are evaluated based on the ground-truth segmentation. Every time a cut candidate is considered, its feature vector is labeled as *valid* or *invalid* by comparing it to the correctly segmented page, as demonstrated in Figure 2. If a cut candidate intersects any region in the correctly labeled page, it is considered invalid; if it does not intersect any region in the correctly labeled page, it is considered valid. The labeled feature vectors are added to the set of training vectors. These assigned labels are also used to determine which cuts should be made in each iteration of the training data generation algorithm.

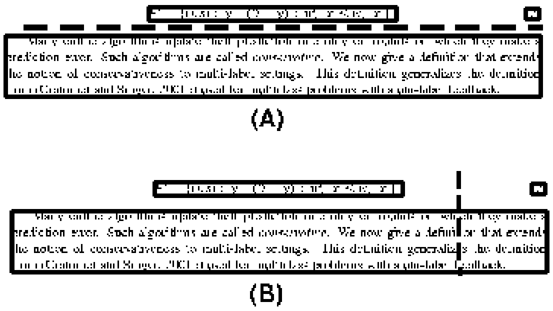

Figure 2. When producing training data, cut candidates are considered valid if they do not intersect any of the hand tagged regions (such as in (A), where the text region can be separated from the equation and equation number above it), and invalid if they do intersect a hand tagged region (such as in (B), where there happens to be a one-pixel wide column of whitespace between letters in the text region).

The features used include the dimensions of the whitespace region created by the cut, the distances between the two ink regions on either side of the cut, and pixel densities from various lines and rectangles sampled from predetermined locations near the cut. These are all computationally inexpensive to calculate.

3.3. Segmentation Performance

Each of the two data sets was separated into training data (about 2/3 of the total), and test data (the remaining 1/3). The algorithms were optimized or trained on training data. Performance was evaluated based on the total number of unmatched segments both on the training and test sets. Two segments *match* if each boundary (top, left, bottom, right) is within 5 pixels of the corresponding boundary of the other segment. An unmatched segment is a predicted or actual segment that does not match any segment from the other set. Results are shown in Table 2.

The Learn-To-Cut algorithm significantly outperformed all three of the benchmark algorithms. The fact that LTC

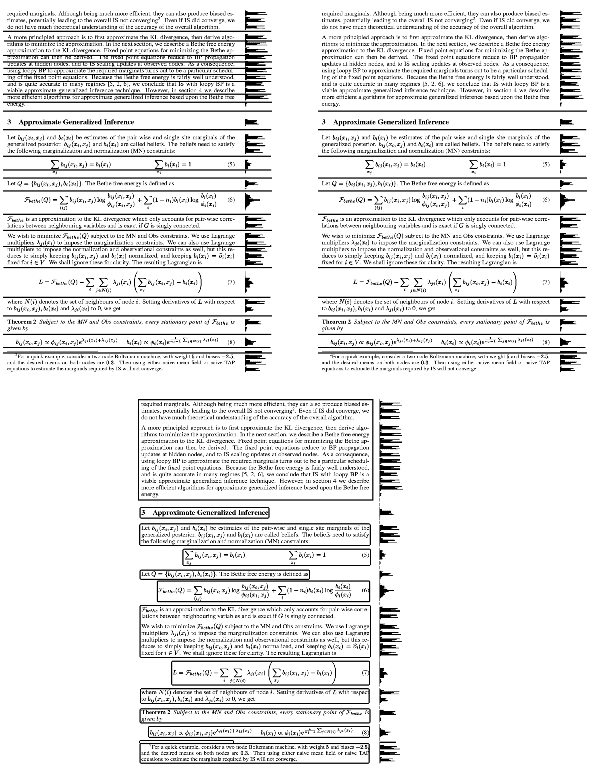

Figure 3. Stages of one iteration of the Concurrent Learn-To-Cut algorithm. It began by using the projection profile of the page to find valleys where the ink density is 0 (upper left). Next, it used a logistic regression classifier to decide which valleys to cut (upper right). The algorithm is applied recursively to the resulting regions (bottom), this time searching for vertical cuts.

outperformed xycut even on the training data, where the optimal values of all parameters for xycut were found, implies that the additional information used by LTC is actually beneficial.

An informal inspection of the unmatched segments in the LTC results suggest two major sources of error. The first is the unique layout of the first page of each document; cutting rules that serve well in the body of the document may yield poor performance on the first page, which generally employs a markedly different layout. The second noted source of error is the algorithm segmenting adjacent regions of the same type either more finely or coarsely than was done by the human tagger. For example, each reference was given its own region by the human tagger, but multiple references were often grouped together by the algorithm. Likewise, consecutive paragraphs were joined by the human tagger, but were often separated by the algorithm.

	Act. segs	xycut	Voronoi	Smear	LTC
NIPS Train	3445	2504	3701	2580	1414
NIPS Test	1473	1200	1535	1103	826
JMLR Train	3521	1228	3228	1439	772
JMLR Test	2085	657	1890	817	536

Table 2. Number of unmatched segments. The first column indicates the number of actual segments in the data set. Note that an error by the algorithm may create multiple unmatched actual and predicted segments.

4. Labeling

The labeling problem is a natural fit to the supervised learning paradigm. Once ink regions have been identified, a classifier can be used to assign the appropriate label, based on a vector of numerical features. An example of a labeled page is shown in Figure 4. The classifier can be trained with examples from the labeled training set. Three algorithms were selected for evaluation: K Nearest Neighbors (KNN), logistic regression (LR), and maximum entropy Markov models (MEMM)[5].

For each ink region, a vector of 59 features describing it is created based on its position, ink density, ink marks contained, and the page number on which it appears. All features were normalized such that they ranged between zero and one. For the MEMM, which requires features to be presented in discrete sequences, the regions on each page were ordered top-to-bottom, left-to-right, in an attempt to approximate their reading order. Each page was treated as a single sequence.

Dataset	KNN	LR	MEMM
NIPS: Training	85.3%	98.3%	99.5%
NIPS: Testing	80.7%	88.1%	87.6%
JMLR: Training	88.8%	97.7%	99.3%
JMLR: Testing	83.6%	90.6%	92.4%

Table 3. Success rates for finegrained classification among 25 precise labels.

The three algorithms were trained and tested using the hand annotated region types as supervised class labels on both the JMLR and NIPS data sets. Success rates in terms of the 25 precise labels, as well as the 4 generalized labels, are shown in Tables 3 and 4. All three algorithms showed strong performance, especially among the 4 generalized categories. As expected, KNN had the weakest performance, and MEMM had the strongest. As another point of comparison, Summers[7] used a complex rule-based system to achieve an average "precise accuracy" of 85.5% among 16

Dataset	KNN	LR	MEMM
NIPS: Training	97.3%	99.8%	99.9%
NIPS: Testing	95.6%	96.0%	96.0%
JMLR: Training	98.1%	99.9%	99.9%
JMLR: Testing	97.8%	99.0%	99.3%

Table 4. Success rates for coarse classification among the 4 generalzed labels.

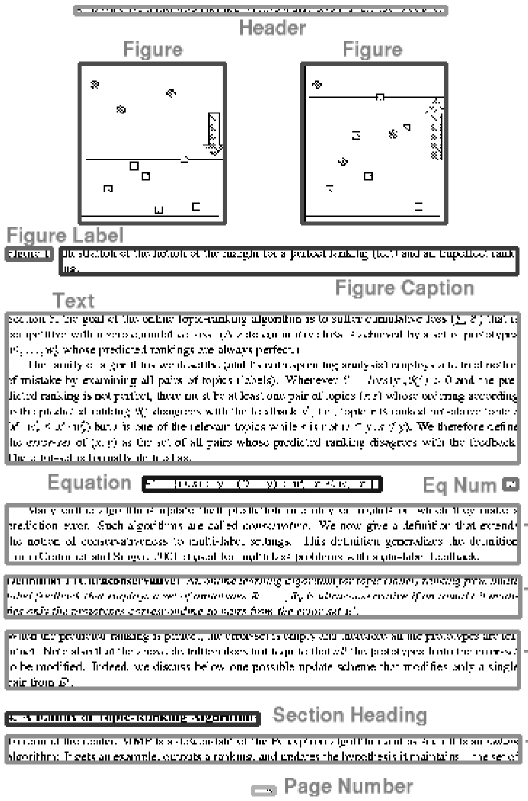

Figure 4. A page image with tagged regions. Bounding boxes are shown, with logical labels added near each region in light gray.

categories, and a "generalized accuracy" of 86.0% on an unspecified (but smaller) number of categories.

Errors were most common for classes with few instances in the training data, such as images, suggesting that not enough examples were present for the algorithms to learn to classify these effectively. The fact that the training error is so much lower than the test error for both Logistic Regression and MEMM also suggests that the use of larger training data sets would improve results. Many of the errors fell into two groups: confusing items of similar classes (such as *section heading* and *subsection heading*), and classifying ambiguous items (such as small equations listed in bullet-form, which could reasonably be classified as *equation* or *bullet item*) differently than the human tagger chose to.

5. Conclusions

Our first attempts at applying established statistical pattern recognition techniques to the problems of page segmentation (physical layout analysis) and labeling (logical layout analysis) have produced very encouraging results. The Concurrent Learn-To-Cut segmentation algorithm outperformed two existing algorithms of similar complexity. All three rudimentary statistical classification algorithms exhibited strong results for the labeling problem. Although direct comparison with other techniques on the same data set was not possible, classification results are comparable or superior to those reported on other data using complex rule-based systems. These results are especially encouraging given that both the classifiers and feature vectors used were relatively simple. Further work is underway involving more powerful learning algorithms and richer feature sets.

References

[1] A. Dengel. Initial learning of document structure. In *Proc. of the Second International Conf. on Document Analysis and Recognition*, volume 20-22, pages 86–90, Oct. 1993.

[2] M. Krishnamoorthy, G. Nagy, S. Seth, and M. Viswanathan. Syntactic segmentation and labelling of digitized pages from technical journals. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(7):737–747, July 1993.

[3] S. Mao and T. Kanungo. Empirical performance evaluation of page segmentation algorithms. In *Proceedings of SPIE Conference on Document Recognition*, San Jose, California, 2000.

[4] S. Mao, A. Rosenfeld, and T. Kanungo. Document structure analysis algorithms: a literature survey. In *Proc. SPIE Electronic Imaging*, volume 5010, pages 197–207, Jan. 2003.

[5] A. Mccallum, D. Freitag, and F. Pereira. Maximum entropy markov models for information extractions and segmentation. In *Proc. 17th International Conf. on Machine Learning.*, 2000.

[6] G. Nagy and S. Seth. Hierarchical representation of optically scanned documents. In *Proceedings of International Conference on Pattern Recognition*, volume 1, pages 347–349, July 1984.

[7] K. Summers. Near-wordless document structure classification. In *Proc. of the Third International Conf. on Document Analysis and Recognition*, pages 426–456, Montreal, Canada, Aug. 1995.