Distinguishing Mathematics Notation from English Text using Computational Geometry

Derek M. Drake
Lehigh University, CSE Dept.
Bethlehem, PA 18015 USA
dmd7@lehigh.edu

Henry S. Baird
Lehigh University, CSE Dept.
Bethlehem, PA 18015 USA
bairdcse.lehigh.edu

Abstract

A trainable method for distinguishing between mathematics notation and natural language (here, English) in images of textlines, using computational geometry methods only with no assistance from symbol recognition, is described. The input to our method is a "neighbor graph" extracted from a bilevel image of an isolated textline by the method of Kise [8]: this is a pruned form of Delaunay triangulation of the set of locations of black connected components. Our method first attempts to classify each vertex and, separately, each edge of the neighbor graph as belonging to math or English; then these results are combined to yield a classification of the entire textline. All three classifiers are automatically trainable. Features for the vertex and edge classifiers were selected semi-manually from a large number in a process driven by training data: this stage is potentially fully automatable. In experiments on images scanned from books and images generated synthetically, this methodology converged in three iterations to a textline classifier with an error rate of less than one percent.

1 Introduction

The capability of distinguishing between different types of content within a page image can improve the performance of document analysis systems. In today's commercial OCR machines, recognition of mathematical notation remains far inferior to recognition of natural language text. In applications where mathematical content is important, it is perhaps better to identify it, isolate it, and protect it from the blundering attentions of the text recognition subsystem. Labeled math subimages could then be presented to readers and browsers of the OCR results as collections of images uncorrupted by OCR errors, for example in "visual math indexes" to assist fast search. Also, since several experimental systems exist to recognize mathematics [10, 5, 3] which run independently of commercial OCR machines, automatic isolation of math images could allow them to be applied selectively where they are most likely to succeed. The Infty system [12] is an example of such an integration.

Mathematics notation is often largely—though seldom entirely—independent of the natural language of the surrounding text, and so there may be important uses for a "language-free" mathematics recognition system applicable independently of any language-specific OCR system.

For these reasons, we suggest that it is interesting to investigate techniques for identifying mathematics notation automatically within document images that can easily be extended to work well within a wide variety of natural languages. This paper reports a step towards this capability: although we report tests only on English language documents, our methodology has been designed to admit easy extension to other languages in three ways:

1. it does not depend on any symbol classifier;
2. it does not depend on prior knowledge of character font, size, or spacing; and
3. it is largely automatically trainable (in particular, it does not rely on any handcoded special case analysis).

The promise of this approach is illustrated by the fact in only three iterations of the methodology—refining the feature set each time—we arrived at a classifier that discriminated between math and textline-images with an error rate of less than one percent.

2 Prior Work

Pioneering methods for recognizing math and text, by Okamoto [10], performed this separation by hand. Later

[0]Accepted for publication in *Proc., IAPR 8th Int'l Conf. on Document Analysis and Recognition*, Seoul, Korea, August 29 – September 1, 2005. [in press]

methods, investigated by Fateman [5], consisted of separating black connected components into two tentatively labeled math/text classes: heuristics assign classes initially and then iterate. This method works with both inline and display math but tends to have trouble with italics and numbers, wrongly assigning them to the math class. Both of these methods depend on recognizers for every character and symbol that can occur. Choudhury *et al* [3] extracted both inline and display math using spatial layout cues (indentation, centering, etc) plus recognition of a dozen or so special symbols: this paper also gives a valuable survey of other related prior work, which will not repeat here for lack of space.

After studying these methods we remained unconvinced that reliance on symbol recognition is necessary or desirable. We observed that mathematics, compared to text, tends to contain characters of more variable sizes, more characters that are positioned diagonally and vertically relative to adjacent characters, and more variable spacing. We conjectured that these peculiar spatial properties might be sufficient cues in the vast majority of cases. We have been influenced in this belief by the success of computational geometry methods [1, 6, 11] on a wide variety of page layout analysis tasks. Kise *et al*'s trailblazing studies [7, 8] proved that an analysis of Delaunay triangulations—and a pruned version of them called "neighbor graphs" (more on this below)—of the locations of black connected components supports accurate and versatile segmentation into textblocks and textlines. Lu *et al* [9] use a similar method using features from the neighbor graph for grouping words.

3 Development of the Math/Text Classifier

We began with existing code, kindly provided by Professor Koichi Kise of Osaka Prefecture University for the algorithms of [7, 8]. Due to lack of space we must refer the reader to his papers for a complete tutorial introduction to the following sketch. Briefly, Kise's algorithms operate on bilevel (black & white) images of documents, extracting from it a set of black connected components (which we will call '*bcc*'s hereafter). An example of such an image is shown in Figure 1. For this set of bcc's, Kise computes a variant of the Delauney triangulation where, in an important innovation, he uses "area" sites where each area consists of the set of black pixels of one bcc, instead of point sites as is usual in the computational geometry literature. Each 'node' in his triangulation represents a single bcc and every bcc is represented by a node. 'Edges' in the triangulation connect two neighboring nodes and therefore capture information about the local relative arrangement of nearby bccs. The triangulation is pruned of redundant edges, giving what Kise calls the "neighbor graph." An example of a neighbor graph, corresponding to the image in Figure 1, is

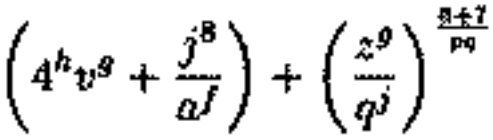

Figure 1. Example of a Bilevel Image of a Math Textline. Note that each character is generally, but not always, a separate black connected component.

Figure 2. Example of Kise's Neighbor Graph of a Math Textline Image. The graph is drawn over the math textline image. Note that each edge of the graph connects two neighboring black connected components.

shown in Figure 2. Both Kise's neighbor graphs and standard point-site Delauney triangulations possess a property that is critically important for our purpose: they are invariant under similarity transformation (translation, scale, and rotation). Thus they do not require prior knowledge of digitizing resolution (spatial sampling rate), type size, position on the page, and skew angle.

We made a few minor modifications to Kise's code:

- we selected a runtime option that includes all small bcc's (the code ignores small bcc's by default);
- we allowed textlines to be arbitrary trees instead of near-linear paths; and
- we modified Kise's 'line extractor' program to write feature vectors for edges and nodes of the neighbor graph.

We extracted features directly from the edges and nodes of the neighbor graph. These features were chosen to capture information about spatial arrangements, especially local relative location, of bcc's. Note that many of these features are sensitive to changes in skew angle (it would be possible in principle to moderate this sensitivity by expressing edges' orientation relative to the overall textline orientation).

For each edge, we computed 29 binary-valued (boolean) features:

1. Feature 1: Shadowing bit – do the two bcc's connected by this edge 'shadow' each other (overlap when projected horizontally)?

2. Features 2–9 (mutually exclusive): Angle classification – bit 2 is set if angle between the two bcc's (bcc1 & bcc2) is in range $[-90, -77.5)$, bit 3 if in range $[-77.5, -55)$,..., bit 9 if $\geq$77.5.

3. 10–19 (mutually exclusive): Area ratio = $\frac{\min(\text{area(bcc1),area(bcc2)})}{\max(\text{area(bcc1),area(bcc2)})}$, bit 10 is set if area ratio is in range $[0.0, 0.1)$, bit 11 set if in range $[0.1, 0.2)$,..., bit 19 set if $\geq$0.9.

4. 20–29 (mutually exclusive): Diameter ratio = $\frac{\min(\text{diameter cc1,diameter cc2})}{\max(\text{diameter(bcc1),diameter(bcc2)})}$, bit 20 set if ratio in range $[0.0, 0.1)$, ..., bit 29 set if $\geq$0.9.

For each node, we computed 77 binary features:

1. 1–13 (mutually exclusive): Aspect Ratio = $\frac{\text{width}}{\text{height}}$, Bit 1 is set if ratio in range $[0.0, 0.25)$,..., bit 13 is set if $\geq$ 3.0.

2. 14–33 (mutually exclusive): Diameter and Area Ratio = $\frac{\text{diameter}}{\text{area}}$. Bit 14 is set if ratio in range $[0.0, 0.05)$,..., bit 33 set if ratio $\geq$ 0.95.

3. 34–44 (mutually exclusive): Fanup = number of nodes to which this node has an edge to that has an angle in the range $[45.0, 135]$. Bit 34 is set if 0,..., bit 44 is set if $\geq$ 10.

4. 45–55 (mutually exclusive): Fandown = number of nodes to which this node has an edge to that has an angle in the range $[225.0, 315.0]$. Bit 45 is set if 0,..., bit 55 is set if $\geq$ 10.

5. 56–66 (mutually exclusive): Fanright = number of nodes to which this node has an edge to that has an angle in the range $[315.0, 45.0]$. Bit 56 is set if 0,..., bit 66 is set if $\geq$ 10.

6. 67–77 (mutually exclusive): Fanleft = number of nodes to which this node has an edge to that has an angle in the range $[135.0, 225.0]$. Bit 67 is set if 0,..., bit 77 is set if $\geq$ 10.

We then trained three classifiers: for edges, nodes, and textlines. Both the node and edge classifiers are quadratic classifiers, while the textline classifier is a simple thresholding classifier. Quadratic features are obtained by Boolean ANDing every pair of binary features. After making the standard assumptions that features are class conditionally independent and that class priors are equally probable, we arrive at the following textbook [4] quadratic classifier:

$$\sum_{k=0}^{n^2-1} \frac{k^3}{2 * k + \alpha} = \frac{\prod_{j=n+2}^{1} \log \gamma^{-2} + \beta}{\delta}$$

Figure 3. Sample Math Textline

The quick brown fox fell in the river after eating too much steak.

Figure 4. Sample Text Textline

Let g($\boldsymbol{x}$) be a discriminant function for either a node or an edge, and let $p_{ij} = P(x_i = 1, x_j = 1|\text{math})$ and $q_{ij} = P(x_i = 1, x_j = 1|\text{text})$. Then

$$g(\boldsymbol{x}) = \sum_{j=1}^{d} \sum_{k=j+1}^{d} \ln \left(\frac{p_{jk}(1 - q_{jk})}{q_{jk}(1 - p_{jk})} \right) x_j x_k + \sum_{k=1}^{d} \ln \left(\frac{1 - p_k}{1 - q_k} \right)$$

where d is the dimensionality of the feature vector $\boldsymbol{x}$. We decide 'math' if $g(\boldsymbol{x}) > 0$ and 'text' otherwise.

The textline classifier is constructed as follows. First, the training set is read in and used to train the classifiers to calculate decision boundaries and estimate class conditional probabilities. Then, the nodes and edges corresponding to a textline are read in and each node and edge is classified as either math or text. The classifiers are combined by thresholding their results and deciding the type of textline by choosing 'math' if the sum of math nodes and math edges is larger than the sum of text nodes and text edges, and deciding 'text' otherwise

4 Experimental Results

The input to both training and testing phases were images of isolated textlines which we cut out of page images manually, or synthesized in isolation using LaTeX. We first trained and tested the edge and node classifiers on the data set described in Table 1. An example of the math and text textlines used is given in Figures 3 and 4.

Tables 2 and 3 show the node and edge classifier's confusion matrices on the training set and Tables 4 and 5 show the confusion matrices for the test set. Tables 6 and 7 show the thresholding classifier's confusion matrices. Finally, Table 8 summarizes the combination of the three classifiers and their error rates. More extensive data testing will be needed to determine if these error rates will generalize.

From Table 8, we can see that even though each of the individual classifiers had relatively poor classification power, the combined thresholding classifier was able to take advantage of the uncorrelated errors of each and thus provide much higher classification power.

In Figure 5 we show the three misclassified textlines with their neighbor graph edges drawn over them. The failures

Data Set: Component Type	Training	Testing
Math Edges	3827	4005
Text Edges	5531	5317
Math Nodes	2273	2269
Text Nodes	5000	4803
Math Textlines	68	68
Text Textlines	64	64

Table 1. Total Edge and Node counts and Total Textline counts for Training and Testing Data Sets

Classified as: True class	Math	Text
Math	1986	287
Text	363	4637

Table 2. Training Set Node Confusion Matrix

Classified as: True class	Math	Text
Math	2668	1159
Text	1121	4410

Table 3. Training Set Edge Confusion Matrix

Classified as: True class	Math	Text
Math	2007	262
Text	359	4444

Table 4. Testing Set Node Confusion Matrix

Classified as: True class	Math	Text
Math	2902	1103
Text	1094	4223

Table 5. Testing Set Edge Confusion Matrix

Classified as: True class	Math	Text
Math	66	2
Text	0	64

Table 6. Training Set Textline Confusion Matrix

Classified as: True class	Math	Text
Math	67	1
Text	0	64

Table 7. Testing Set Textline Confusion Matrix

Data Set: Component Type	Training	Testing
Math Textlines	0.029	0.015
Text Textlines	0.000	0.000
Textlines Overall	0.015	0.008

Table 8. Textline Error Rates

apparently occurred because all three are largely horizontal text with very few transitions above and below the textline. That we make fewer errors on the training set than on the test set is perhaps an artifact of the small sample size.

The CPU runtime required for classification is modest. On a SunBlade 150 (UltrasSparc IIe) 650 MHz machine, classifiying a test image of textline required 0.075 CPU seconds on average. The code has not been optimized for speed. The asymptotic runtime requiremenats are also low: to build the neighbor graph approximately $O(nlogn)$ where n is the no. of bcc's and of course constant time per edge and per vertex to classify.

5 Conclusions & Future Work

In the test on 132 images of math and text lines, only one error was seen, for an error rate less than 0.76%; at 95% statistical confidence, this implies an error rate less than 3%. Computational geometry analysis of page layout thus appears to be a promising way to discriminate between mathematics notation and natural language text within images of textlines.

Although so far we have tested only English-language documents, we anticipate that our methodology will admit easy extension to other languages, due to these three properties:

1. it does not depend on any symbol classifier;
2. it does not rely on prior knowledge of font, size, or spacing; and
3. it is largely automatically trainable.

It may be possible to correct many of the mistakes we have seen by adding a few features that, like the ones we have have used so far, capture purely spatial relationships

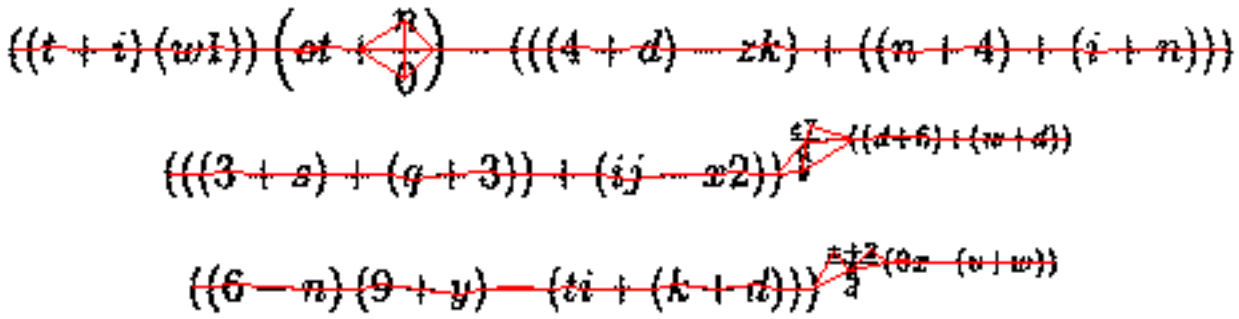

Figure 5. The three incorrectly classified textlines: all, of course, are math textlines wrongly classified as English text. The first two images are from the training set of 132 textline images, while the last is from the test set of 132 images. We have drawn, over the text, the neighbor graph for each.

among black connected components (bcc's): the distributions of horizontal spacing between bcc's, and the distribution of heights of bcc's, both seem promising.

We are interested also in applying our classifier to the problem of locating "inline" math expressions (*i.e.* mixed with text within the same textline). Ultimately we'd like to extend the approach to assist in the detection of textlines themselves, whether they are dominantly math or text, or a mixture.

References

[1] H. S. Baird, "Background Structure in Document Images," in H. Bunke, P. S. P. Wang, & H. S. Baird (Eds.), *Document Image Analysis*, World Scientific, Singapore, pp. 17-34, 1994.

[2] R. Cattoni, T. Coianiz, S. Messelodi, C. M. Modena, "Geometric Layout Analysis Techniques for Document Image Understanding: a Review," *Technical report, IRST*, Trento, Italy, 1998.

[3] S. P. Chowdhury, S. Mandal, A. K. Das, B. Chanda, "Automatic Segmentation of Math-Zones from Document Images," *Proc., IAPR 7th ICDAR*, Edinburgh, Scotland, August 4–6, 2003.

[4] R. Duda, P. Hart, D. Stork, *Pattern Classification* (2nd Ed.), John Wiley & Sons, October 2000.

[5] R. Fateman, "How to find Mathematics on a Scanned Page," Proc., SPIE, Vol. 3967, pp. 98–109, 1999.

[6] D. Ittner, H. S. Baird, "Language-Free Layout Analysis," *Proc., IAPR 2nd Int'l Conf. on Document Analysis & Recognition*, Tsukuba Science City, Japan, pp. 336-340, October 1993.

[7] K. Kise, M. Iwata, K. Matsumoto, "On the Application of Voronoi Diagrams to Page Segmentation," *Proc., Workshop on Document Layout Interpretation and its Applications (DLIA99)*, Bangalore, India, September 18, 1999.

[8] K. Kise, A. Sato, M. Iwata, "Segmentation of Page Images using the Area Voronoi Diagram," *Computer Vision and Image Understanding*, *70*, 1998, 370–382.

[9] Y. Lu, Z. Wang, C. L. Tan, "Word Grouping in Document Images Based on Voronoi Tessellation," *International Workshop on Document Analysis Systems (DAS)*, Florence, Italy, Sept 8–10 2004.

[10] M. Okamoto, A. Miyazawa, "An experimental implementation of a document recognition system for papers containing mathematical expressions," *Structured Document Image Analysis*, Springer-Verlag, 1992, 36–51.

[11] G. Toussaint, "Computational Geometry for Document Analysis," *Proc., 3rd Annual Symp. Document Analysis and Information Retrieval*, pp. 23–42, 1994.

[12] M. Suzuki, T. Kanahori, N. Ohtake, and K. Yamaguchi, "An Integrated OCR Software for Mathematical Documents and Its Output with Accessibility," in *Blind People: Access to Mathematics*, Springer-Verlag (LNCS Vol. 3118), 2004.