Extraction, layout analysis and classification of diagrams in PDF documents

Robert P. Futrelle, Mingyan Shao, Chris Cieslik and Andrea Elaina Grimes
College of Computer and Information Science
Northeastern University
Boston, MA 02115
{futrelle,myshao,ccieslik,agrimes}@ccs.neu.edu

Abstract

Diagrams are a critical part of virtually all scientific and technical documents. Analyzing diagrams will be important for building comprehensive document retrieval systems. This paper focuses on the extraction and classification of diagrams from PDF documents. We study diagrams available in vector (not raster) format in online research papers.

PDF files are parsed and their vector graphics components installed in a spatial index. Subdiagrams are found by analyzing white space gaps. A set of statistics is generated for each diagram, e.g., the number of horizontal lines and vertical lines. The statistics form a feature vector description of the diagram. The vectors are used in a kernel-based machine learning system (Support Vector Machine). Separating a set of bar graphs from non-bar-graphs gathered from 20,000 biology research papers gave a classification accuracy of 91.7%. The approach is directly applicable to diagrams vectorized from images.

1. Introduction

Documents in Portable Document Format, PDF [1] allow sophisticated formatting but can have complex internal structure. Analysis of their components and layout can be daunting. Though there is a great deal of work on the analysis, indexing and retrieval of the text content of documents in many formats, including PDF, much less attention has been paid to the graphics content of documents. In particular, it would be quite useful to analyze the internal structure of diagrams to extract their content. This would allow them to be classified for indexing and retrieval purposes. The overwhelming majority of work on document analysis today is focused on document image analysis. Many millions of documents now have their text available in electronic form, obviating the need for OCR. But the figures in these documents still require analysis if we are to be able to index and retrieve them in comprehensive document systems. The diagrams (line drawings) are available in vector format in some documents, e.g., PDF files, or more commonly, in raster format, e.g., GIF and JPEG. This paper focuses on the classification of vector-based diagrams. We have chosen all of the diagrams contained in a collection of 20,000 biology research papers in PDF format.

For some time, our group has been involved with the analysis of diagrams in technical documents to a level of detail that includes parsing diagrams to produce syntactic descriptions for them [2-5]. This paper reports on extensions of our work to PDF documents, including using supervised machine learning (kernel methods) to classify diagrams.

The stages of analysis are:

1. PDF documents are downloaded from a large collection, American Soc. Microbiology (ASM)
2. The PDF command sequences comprising each document are parsed, producing a corresponding object sequence.
3. Using an interpreter, the command sequence is analyzed to discover visible objects.
4. The resulting objects are rendered into coarse 1D and 2D spatial indexes (SPAS).
5. Multiple diagrams and subdiagrams on each page are recursively extracted by looking for horizontal and vertical separating white space bands.
6. Statistics on the diagrams are compiled (number of lines, curves, text items, etc.)
7. The statistics are used to train a Support Vector Machine to classify bar graphs versus non-bar-graphs.

The strategy above has been successful, yielding 91.7% overall classification accuracy (leave-one-out measure) in these initial studies. It is important to emphasize that the PDF documents discussed here that contain vector graphics are in the distinct minority, since most PDFs we have examined contain figures in raster format. In our separate *Strategic Vectorization Project*, we are developing a new approach to converting raster images of diagrams to vector format (vectorization) (Crispell and Futrelle, in preparation). The raster images are obtained in JPEG or GIF format from the HTML versions of papers. In our corpus of 50,000 ASM HTML papers there are approximately 500,000 such raster-

Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR'03)
0-7695-1960-1/03 $17.00 © 2003 IEEE

format diagrams (counting the multiple diagrams that frequently occur together in a single figure). In the 20,000 PDF documents in the collection, totaling about 120,000 pages, we found only 52 pages that had vector-based diagrams. Though this is a tiny percentage, it offered us the opportunity to experiment with classification methods which we will later apply to the much more numerous vectorized diagrams. Full syntactic analysis of vector diagrams has been discussed in our previous papers [2, 3, 5] and is not covered here.

2. Extracting vector objects

The PDF documents were downloaded from the American Society for Microbiology site, asmusa.org, using a web-bot built on the Java HTTPClient package [6]. (Northeastern University has a license agreement with the ASM to download and analyze six years worth of their publications.) The PDF commands are parsed using Etymon PJ written in Java [7]. This produces a vector of *graphics objects* (GOs), Java class instances, that are in one-to-one correspondence with the sequence of PDF commands in the document. The objects are then converted to our own GO representation to make them independent of the PJ classes. An excerpt from such a PDF sequence describing a *path* is,

```
172.443 482.809 m
172.443 483.756 l
172.885 483.775 173.187 483.805 173.365 483.844 c
173.646 483.906 173.871 484.027 174.049 484.215 c
```

which produces four GOs, corresponding to a move of the drawing position (m), drawing a line from there to the position given (l) and then drawing two curved lines in succession (c). PDF also provides a rectangle object, as a primitive. The most complex object that we have to deal with is the PDF *Graphics State*. It encapsulates rendering parameters such as line widths, fill color, font face, font size, and the affine transformation that is applied during rendering, e.g., to reposition the page coordinate system or to rotate text, typically by 90°.

The initial identification of pages that contain graphics has been done by counting the number of GOs on a page. If a page has > 20 GOs, it is likely to be a diagram and was analyzed further.

3. The complexities of PDF rendering

PDF commands involve more than simply building paths and stroking or filling them. Changes to the Graphics State are also made, applied to the current Graphics State. The identity of the current Graphics State can be changed by pushing and popping it from a stack built into PDF. We discovered that PDF rectangles are used for two distinct purposes. One is the normal use to create visible outlined and/or filled rectangular regions. The other is to define clipping regions that restrict what portions of stroked or filled paths are visible. Some clipping regions are defined by more complex paths such as the one in Sec. 2. The clipping paths don't directly produce graphics that is visible to the human reader of an article and are not counted in characterizing the visible diagram. We discovered that in the diagrams we collected that the clipping regions could be safely ignored in analyzing the visible portions of the diagrams because they have little or no effect on its appearance.

When the GOs are isolated, they, plus their attendant clipping regions and Graphics State commands are used, via the Etymon PJ tools, to create a PDF page that contains only the graphics. The running text, footnotes, etc., are not included. This graphics-only page can be examined and printed using the Adobe Acrobat Reader.

In order to discover what objects were visible, we had to track the Graphics State stack. We constructed a specialized PDF interpreter, acting on GOs, that collected paths, including lines, curves and rectangles and processed them in two ways. If they were subsequently found to define a clipping region, or if they were stroked and filled with white, the paths were discarded. If they were stroked and/or filled with any non-white color, they were added to a list of visible objects. Only these visible objects were output by the interpreter and used for the diagram analyses that followed.

4. Installing objects in a spatial index

In order to analyze the spatial layout structure of the graphics in a page, a spatial index is used. The one used is SPAS (Spatially Associative Substrate) that we developed for diagram parsing [2-4].

A 64 cell wide by 128 cell tall array was used, a coarse representation of the x,y page space. Each array element contains a *Cell* object, a Java instance. The GOs in a page are rendered at the appropriate resolution and for each Cell through which the object passes, a reference to the GO is added to a list in the corresponding Cell as shown in Fig. 1. A pair of one-dimensional projection indexes are built as shown in the figure. The contents of these *ProjectionCells* in the 1D arrays are the merged contents of all Cells in the 2D array at that position. For example, a ProjectionCell at a specific x index contains information from the Cells in the 2D array at that x index and at all y indexes. A ProjectionCell in the 1D array contains the set of all GOs at that 1D position, and in addition, the total number of straight lines, rectangles, curves and text objects at that position.

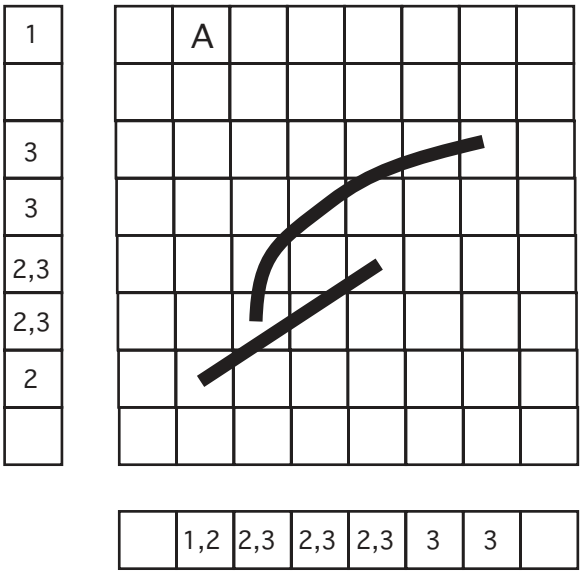

Figure 1. An 8x8 SPAS structure for spatial indexing. The paths for each object are installed in (rendered into) the SPAS and the cell sets for each path are collected. The three objects shown here are "A" (1), a straight line (2) and a curve (3). References to each object are installed in the cells they occupy in the 2D array (ref. numbers not shown here in the 2D array) as well as in the two 1D projection arrays. Additional information and statistics are collected in the ProjectionCells in the 1D arrays.

5. Slice & Dice for subdiagram discovery

For reasons of space economy and coherence of presentation, diagrams in papers often *composite*, containing a number of subdiagrams. We attempt to separate out each subdiagram and then do further analysis on each separately. The separation is done by recursively constructing and examining the 1D SPAS arrays for white space gaps. The SPAS for the entire page is pushed on a stack, popped and examined for any large gaps on the y-axis, typically gaps of 20 points or more (about two font heights). If any are found, new SPAS containing the GOs in the separate y regions are created and pushed on the stack. Each is then examined for white space gaps along the x-axis. The process continues using the stack to handle the inherent recursion. All SPAS discovered along the way that cannot be split are saved for the further processing stages. We call this analysis process "Slice & Dice". An example is shown in Fig. 2. The version of Slice & Dice described above is based on the number of GOs in each 1D SPAS array ProjectionCell. A white space is indicated by a zero count in a ProjectionCell. Extensions to this approach are obvious and will be necessary to achieve the most accurate identification of subdiagrams. For example rows or columns of text or numerical labels are positioned along the edges of diagrams with some white space separating them from the diagram graphics (lines, etc.). It would be an error to identify such a row or column of text as a complete subfigure on its own. So a more refined Slice & Dice algorithm will need to keep such text bound to the adjacent graphics items. Ambiguities can arise when text is positioned between two collections of graphics and a decision needs to be made as to which graphics items will be chosen as the ones bound to the text. We have previously discussed ambiguity in diagram organization in another paper [8], where we suggest ambiguity resolution methods such as context restrictions, semantic restrictions and minimal complexity measures. See also [9]. As one example of a disambiguation strategy, note that the diagram in Fig. 2 has two subfigure labels, "a" and "b", which could aid the Slice and Dice decisions. Our own Fig. 1 has significant white space within it and could require careful analysis.

6. Machine learning for classification

There are many compelling reasons to separate diagrams into various classes. The primary one is to aid in the indexing of diagrams contained in documents which in turn can lead to more powerful and useful retrieval systems. In our work, classification can be used as an aid in deciding which specific grammars are most applicable for parsing a diagram. Classification is a complex topic but is basically divided into two types of methods, *supervised* and *unsupervised*. A typical unsupervised technique is *clustering* in which the algorithm attempts to find a partition of a set of items into classes which contain items that are similar to one another but not as similar to items in other classes [11]. This approach has proved useful in word-meaning classification [12] and will be pursued for diagrams also. In this paper we describe the techniques we have used for supervised learning. For supervised learning, our system was trained by presenting it with instances whose classes were labeled manually. For our initial study we used the complete set of 129 subdiagrams, 65 of which were bar graphs and 64 of which were other, primarily ordinary data graphs, but a few gene diagrams were included. We used a binary classifier, a kernel-based *Support Vector Machine (SVM)*, SVM^{light} [13, 14]. Support vector methods tend to generalize well when classifying new items that were not used in the training; they avoid the over-training problem. The strategy used was "one-against-all", in our case bar graphs versus all non-bar-graphs.

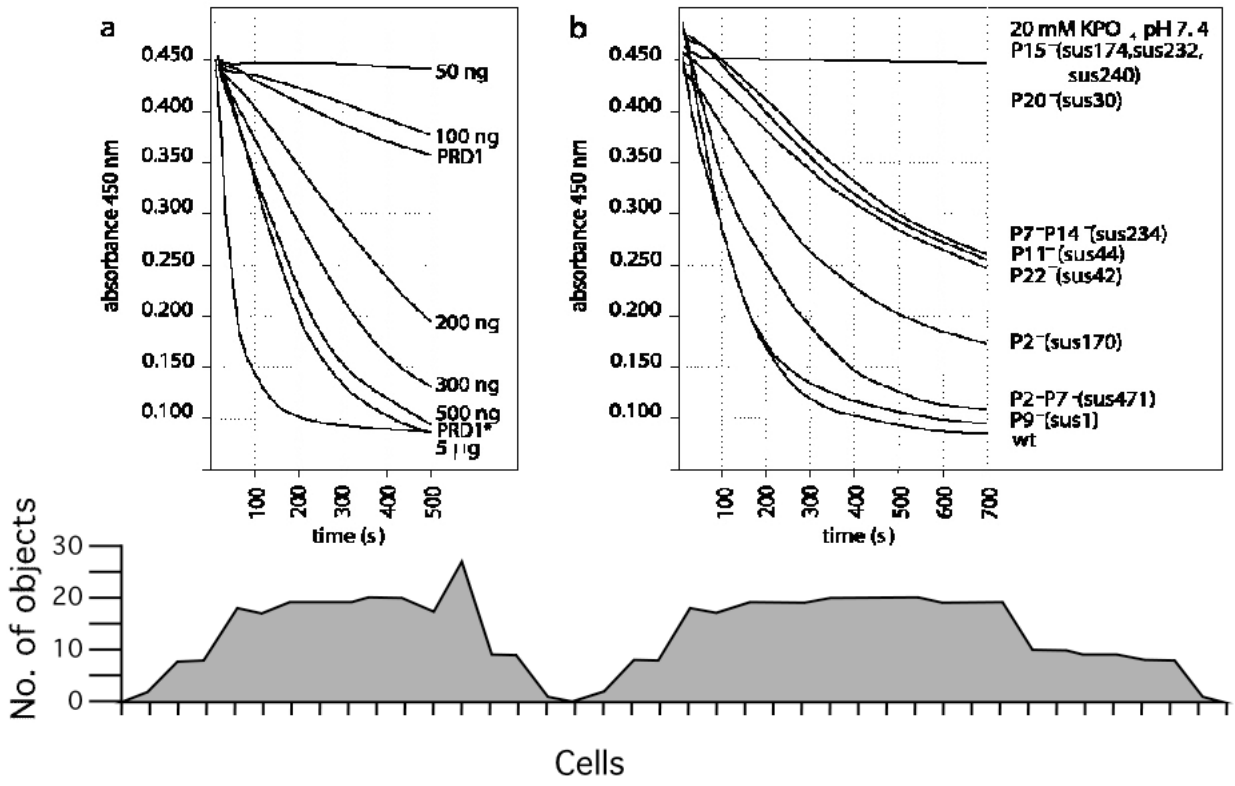

Figure 2. Slice and Dice: Object-count-based signature for a vector format diagram. The signature function below the diagram plots the number of graphic objects directly above the x-cells shown. The horizontal gap between the two subdiagrams shows up clearly as an empty cell, count = 0. Note that this analysis is entirely different from pixel density counts for a raster image. The approach is extended in the machine learning study later in this paper, in which heterogeneous signatures are computed using statistics such as the number of horizontal lines, the number of text objects near small vertical lines, etc. (A "crisp" vector-based PDF file of the diagram image above is available at http://www.ccs.neu.edu/home/futrelle/papers/104-7-3.pdf and the full paper is [10].)

7. Results

As with virtually all machine learning methods, the items to be classified are represented by a vector. We used a vector with the following six integer components which counted various classes of items in each subdiagram:

1. # of rectangles
2. # of datapoints
3. # of lines or curves
4. # of categorial labels (non-numeric)
5. # of short horizontal lines near a long vertical
6. # of short vertical lines near a long horizontal

For the SVM computations, we defined a simple inner-product *kernel*, $K(\mathbf{x_i},\mathbf{x_j})$, for each pair of 6-dimensional vectors $\mathbf{x_i}$ and $\mathbf{x_j}$ as the inner product $\mathbf{x_i} \bullet \mathbf{x_j}$. We also defined a *radial basis function* kernel [13-15] of the form $\exp(-\gamma||\mathbf{x_i}-\mathbf{x_j}.||^2)$. Both kernels performed adequately. The results from the inner-product kernel are reported here. The binary classifier was trained to distinguish bar graphs primarily from ordinary data graphs (which use points and data lines). The bar graphs were labeled as the positive examples.

The overall accuracy was 91.7% = (number of correctly classified diagrams / (total number, 129)). The recall was 98.5% = (number of correct positives / (total positives, 65)). The precision was 85.3% = (number of correct positives / (number classified as positive, 75)).

8. Discussion

8.1. Document image analysis

There is a large amount of work on document *image analysis*, analysis of raster format page images [16, 17]. Many of the approaches to document image analysis are quite applicable to the analysis of vector-based documents

and their included graphics. Among the more closely related papers, we mention one on document zone content classification [18]. In that work 25 parameters are extracted for each rectangular zone on a page, including run length statistics, text glyph features, etc. The approach is tested on the UWCDROM-III document image database [19]. Since the study, as do all studies of document images, deals with pixels in a raster image, no statistics involving classes of graphic objects such as lines or curves can be computed. There can be no later stage of parsing either, as there are no objects to input to a parser. Another study in which a large collection of document metrics was developed is [20], which focused on metrics for images in documents.

Other recent work on comprehensive approaches to document image structure analysis includes [21] and [22].

8.2. PDF document analysis

There are a few studies of PDF document structure. One used the PDFEdit facilities in the Adobe SDK, but it did not appear to produce consistent bounding boxes for the identical documents converted in various ways [23]. Another approach, the AIDAS system[24], used a bottom-up iterative chunking procedure to discover the document logical structure (sections, paragraphs, etc.) based on cues such as font size and style, e.g., large and bold text. The initial conversion of PDF in AIDIS is built on top of the Xpdf tools. Our interpreter, implemented in Java, is focused on extracting the visible components relevant to compiling statistics for machine learning. Being written in Java, it runs on a wide variety of systems and needs no special porting. The same applies to out download and initial parsing using the Etomyn PJ tools, also written entirely in Java. In AIDIS, no detailed analysis of diagram internals was done. Our approach, using a spatial index, makes it possible for us to discover the correct spatial extent of any graphic object including the cubic Bézier curves in PDF. Our SPAS structure includes the detailed location of the graphic elements, at a coarse resolution, rather than just the bounding boxes [23].

8.3. Vectorization of diagrams

The majority of the PDF versions of electronic journal papers we have examined do not contain diagrams in vector form, but in raster form. But the development of techniques to deal classify and otherwise analyze vector diagrams is very important, because it is possible to convert the millions of published diagrams in raster format to vector format using *vectorization algorithms* [25, 26]. Vectorization uses a specialized collection of image processing tools to discover and characterize elements such as lines and curves and generate compact vector representations for them. Our lab has a major research project devoted to vectorization, mentioned earlier.

8.4. Slice and Dice

We have already discussed the problem of ambiguity that can make it difficult to accurately identify subdiagrams. Even an accurate separation can lead to later problems. For example, it is not unusual to have a pair of side-by-side data graphs in which the y-axis label is only given for the graph on the left and assumed to be the same for the graph on the right [27]. Sometimes even the numerical scale values are given only on the left of a pair. So separating them could lead to a loss of information. The grammars we have developed for diagram parsing can deal with such cases [27]. There is a tension between the two extremes -- simply relying on white space for separation versus a detailed parsing-based analysis of each figure taken as a whole. The conclusion is that the process of separation into subdiagrams and detailed diagram analysis and may need to be coordinated in some way and not assumed to be made up of a pair of independent steps.

8.5. Signatures for slicing and classifying

Our Slice and Dice techniques for separating out subfigures use signatures built from simple object counts, Fig. 2. One complication that was not mentioned earlier is the following. In some PDF documents, text characters are themselves rendered by a succession of vector commands, sometimes requiring as many as twenty commands (lines and curves) to render a single character. Blindly counting such commands could badly skew a representation of the diagram content. In fact, in the diagram of Fig. 2 this was the case. Since we have not yet developed techniques for collapsing these command sequences into single character representations, the contribution of text items to the signature in Fig. 2 was computed manually.

The object-based formulation allows us to create more complex structures that can aid in future work on subfigure separation. For example, a feature could be constructed that consists of all short lines adjacent and perpendicular to a long line, and in addition, sets of labels adjacent to the short lines. This could identify a scale structure that could be retained as a unit in the Slice and Dice work.

One of the more powerful aspects of our approach is that if some particular object, O_1, is identified in an X projection index array, the identical object can be located in the Y array and a correlated analysis can be done. This is impossible when pixels alone are projected (unless each pixel is given its own location or object status in the projections!).

8.6. Machine learning for classification

The kernel methods used in SVM-based machine learning [13, 15, 28-30] are free of the "curse of dimensionality' inherent in other methods. That is because the dimensionality is related to the number of training example pairs and not to the intrinsic dimensionality of each training example. So it is possible to use large feature sets for SVM learning. In future work, this might obviate the need for detailed parsing-level analysis. There are two other important advantages of SVM techniques. The theory behind SVMs, *computational learning theory* [30], was designed to avoid overfitting to achieve the highest quality of generalization to as yet unseen examples. In addition, kernel methods allow complex nonlinear mappings of the feature space, in some cases equivalent to infinite-dimensional representations, while retaining a low order of complexity in the actual computations. We have primarily used the simplest inner-product kernels in this work; radial basis function kernels gave similar results.

A major problem that arises in all machine learning systems, and in document understanding in particular [31], is attempting to decide on the class of the document for training purposes. A more detailed analysis has to determine whether or not the structural elements discovered are correct, or even that there is a genuine ambiguity [8] which can confound manual training decisions. At a deeper level work has to be done to understand the class structure and ontology of diagrams. For example, some of the diagrams in our collection are hybrids, in which a single data graph contains both bar data and point and line data.

A major application of machine learning to document analysis is the WISDOM++ system [32]. In that paper, document structure knowledge is represented by means of decision trees and first order rules automatically generated from a set of training documents. Decision trees require careful construction to avoid overfitting, a problem that is largely avoided by support vector machines.

The full characterization of a diagram, its class and its role in the document must also involve the text in the diagram, in the figure caption and in the body of the document where the diagram is discussed. Diagrams and text can be strongly interwoven so that the full content can only be understood by an integrated analysis of the two [33, 34]. Machine learning techniques, and SVMs in particular [13], have been used for text classification and could readily be used to classify diagrams using their graphics content and associated text simultaneously.

Acknowledgements

The work of Dan Crispell, Mike Preshman and Jing Shan was helpful in this project. Supported in part by NSF DBI-0211047, IIS-9978004 and the Northeastern University Institute for Complex Scientific Software, http://www.icss.neu.edu/

References

[1] Adobe Systems Incorporated, *PDF Reference: Version 1.4*, 3rd ed: Addison-Wesley Pub Co, 2001.

[2] R. P. Futrelle, "Strategies for Diagram Understanding: Object/Spatial Data Structures, Animate Vision and Generalized Equivalence," in *10th ICPR*: IEEE Press, 1990, pp. 403-408.

[3] R. P. Futrelle and N. Nikolakis, "Efficient Analysis of Complex Diagrams using Constraint-Based Parsing," in *ICDAR-95 (Intl. Conf. on Document Analysis & Recognition)*. Montreal, Canada, 1995, pp. 782-790.

[4] R. P. Futrelle, "The Diagram Understanding System Demonstration Site (http://www.ccs.neu.edu/home/futrelle/diagrams/demo-10-98/)," 1998.

[5] N. Nikolakis, "Diagram Analysis using Equivalence and Constraints," in *College of Computer Science*. Boston, MA: Northeastern University, 1996, pp. 198.

[6] R. Tschalär, "HTTP Client 0.3-3 (http://www.innovation.ch/java/HTTPClient/)," vol. 2002, 2002.

[7] Etymon Systems, " PJ Classic (http://www.etymon.com/pjc/)," Etymon Systems, 2002.

[8] R. P. Futrelle, "Ambiguity in Visual Language Theory and its Role in Diagram Parsing," in *IEEE Symposium on Visual Languages, VL99*. Tokyo: IEEE Computer Soc., 1999, pp. 172-175.

[9] M. Shilman, H. Pasula, S. Russell, and R. Newton, "Statistical Visual Language Models for Ink Parsing," in *Proc. AAAI Spring Symposium on Sketch Understanding*: AAAI, 2002.

[10] P. S. Rydman and D. H. Bamford, "The Lytic Enzyme of Bacteriophage PRD1 is Associated with the Viral Membrane," *J. Bacteriology*, vol. 184, pp. 104–110.

[11] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. New York: Wiley, 2001.

[12] S. Gauch and R. P. Futrelle, "Experiments in Automatic Word Class and Word Sense Identification for

Information Retrieval," in *Third Annual Symposium on Document Analysis and Information Retrieval*, 1993, pp. 425-434.

[13] T. Joachims, *Learning to Classify Text Using Support Vector Machines*: Kluwer Academic Publishers, 2002.

[14] T. Joachims, "SVMlight Support Vector Machine (http://svmlight.joachims.org/)," 2002.

[15] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge, UK: Cambridge University Press, 2000.

[16] G. Nagy, "Twenty Years of Document Image Analysis in PAMI," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 38-62, 2000.

[17] L. O'Gorman and R. Kasturi, "Document Image Analysis," IEEE Computer Soc. Press, 1997.

[18] Y. Wang, I. T. Phillips, and R. M. Haralick, "A Study on the Document Zone Content Classification Problem," in *Document Analysis Systems (DAS 2002)*, vol. LNCS 2423, D. Lopresti, J. Hu, and R. Kashi, Eds. Berlin: Springer-Verlag, 2002, pp. 212–223.

[19] I. Phillips, "Users' reference manual. CD-ROM, UW-III Document Image Database-III," 1995.

[20] V. Athitsos, M. J. Swain, and C. Frankel, "Distinguishing Photographs and Graphics on the World Wide Web," in *IEEE Workshop on Content-Based Access of Image and Video Libraries*, 1997, pp. 10-17.

[21] M. Aiello, C. Monz, L. Todoran, and M. Worring, "Document Understanding for a Broad Class of Documents," *Intl J. Document Analysis and Recognition*, vol. 5, pp. 1-16, 2002.

[22] S. Klink, A. Dengel, and T. Kieninger, "Document Structure Analysis Based on Layout and Textual Features," in *DAS - International Conference of Document Analysis Systems*. Rio de Janeiro, Brazil, 2000, pp. 99-111.

[23] H. Chao, G. Beretta, and H. Sang, "PDF Document Layout Study with Page Elements and Bounding Boxes," in *Workshop on Document Layout Interpretation and its Applications (DLIA2001)*, 2001.

[24] A. Anjewierden, "AIDAS: Incremental Logical Structure Discovery in PDF Documents," presented at 6th International Conference on Document Analysis and Recognition (ICDAR 2001), Seattle, WA, 2001.

[25] D. Blostein and Y.-B. Kwon, "Graphics Recognition Algorithms and Applications 4th International Workshop, GREC 2001, Kingston, Ontario, Canada, September 7-8, 2001. Selected Papers," Springer Verlag, 2002, pp. 367.

[26] S. Ablameyko and T. Pridmore, *Machine Interpretation of Line Drawings: Technical Drawings, Maps and Diagrams*. London: Springer-Verlag, 2000.

[27] R. P. Futrelle, "Objects in the image can be members of multiple parsed structures [http://www.ccs.neu.edu/home/futrelle/diagrams/demo-10-98/sharing.html]," 1998.

[28] C. J. C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," *Data Mining and Knowledge Discovery*, vol. 2, pp. 121-167, 1998.

[29] H. Byun and S.-W. Lee, "Applications of Support Vector Machines for Pattern Recognition: A Survey," in *SVM 2002*, vol. LNCS 2388, S.-W. Lee and A. Verri, Eds. Berlin: Springer, 2002, pp. 213-236.

[30] V. N. Vapnik, *Statistical Learning Theory*. New York: Wiley-Interscience, 1998.

[31] D. Lopresti and G. Nagy, "Issues in Ground-Truthing Graphic Documents," in *Graphics Recognition Algorithms and Applications 4th International Workshop, GREC 2001, Kingston, Ontario, Canada, September 7-8, 2001. Selected Papers*, D. Blostein and Y.-B. Kwon, Eds. Berlin: Springer, 2002, pp. 46-67.

[32] F. Esposito, D. Malerba, and F. A. Lisi, "Machine Learning for Intelligent Processing of Printed Documents," *J. Intelligent Information Systems*, vol. 14, pp. 175–198, 2000.

[33] R. P. Futrelle and A. Rumshisky, "Discourse Structure of Text-Graphics Documents," in *1st International Symposium on Smart Graphics*. Hawthorne, NY: ACM, 2001.

[34] Y. Watanabe and M. Nagao, "Diagram Understanding Using Integration of Layout Information and Textual Information," in *COLING-ACL*, 1998, pp. 1374-1380.