First publ. in: SAC '10 Proceedings of the 2010 ACM Symposium on Applied Computing, 2010. pp. 8-12

Enhancing Document Structure Analysis using Visual Analytics

Andreas Stoffel
Siemens AG, Germany
andreas.stoffel.ext@siemens.com

Henrik Kinnemann
Siemens AG, Germany
henrik.kinnemann@siemens.com

David Spretke
University of Konstanz, Germany
david.spretke@uni-konstanz.de

Daniel A. Keim
University of Konstanz, Germany
daniel.keim@uni-konstanz.de

ABSTRACT

During the last decade national archives, libraries, museums and companies started to make their records, books and files electronically available. In order to allow efficient access of this information, the content of the documents must be stored in database and information retrieval systems. State-of-the-art indexing techniques mostly rely on the information explicitly available in the text portions of documents. Documents usually contain a significant amount of implicit information such as their logical structure which is not directly accessible (unless the documents are available as well-structured XML-files) and is therefore not used in the search process. In this paper, a new approach for analyzing the logical structure of text documents is presented. The problem of state-of-the-art methods is that they have been developed for a particular type of documents and can only handle documents of that type. In most cases, adaptation and re-training for a different document type is not possible. Our proposed method allows an efficient and effective adaptation of the structure analysis process by combining state-of-the-art machine learning with novel interactive visualization techniques, allowing a quick adaptation of the structure analysis process to unknown document classes and new tasks without requiring a predefined training set.

Categories and Subject Descriptors

I.7.5 [**Computing Methodologies**]: Document Capture—*Document analysis*

General Terms

Automatic Document Structure Analysis, Visual Analytics

1. INTRODUCTION

Libraries, national archives and companies are faced with huge amount of documents that are shelved in archives. The archives are full of images, books, file cards and other documents. Making these cultural assets and documents available to a broader public and allowing an efficient search and retrieval of information raised the desire to make these documents available in electronic form, which resulted in several mass digitization projects worldwide.

Searching and information retrieval for text documents is a well known task. Traditionally, the bag-of-word model is used for indexing purposes, which does not consider the position of the words in the documents [8]. Augmenting the bag-of-word models with additional information about the logical structure of the documents would allow more expressive queries for retrieval purposes. For instance, a query could be narrowed to a specific logical part of the document, like "introduction: document engineering" to search for documents that contain the terms "document" and "engineering" in the introduction.

Challenges for structure analysis tasks are heterogeneous document collections with many different document types that may also change over time. Manually adapting the structure analysis process to each document type is a laborious task and maybe uneconomical. The proposed structure analysis system addresses this problem by reducing the manual adaption costs using a combination of machine learning algorithms with manual verification and correction of the structure information. The machine learning algorithm learns directly from the user's input and adapts itself stepwise to new document types.

2. RELATED WORK

The analysis of the document structure is mainly used for document image analysis and information extraction. Rule-based approaches are basic techniques which evaluate predefined rules to assign labels to the text regions [6, 7, 10]. Alternatively, various kinds of grammars have been proposed for structure analysis [1, 3, 13]. These systems model documents with different kinds of grammars and assigns labels to text regions by applying the grammar rules to the documents. Other structure analysis techniques include, for example, emergent computing [5] and n-grams [2]. Overviews of structure analysis approaches for document images can be found in [9, 11]. All mentioned approaches have in common that they are developed for a specific task and document type. Using any of the presented method for a different task, would mean to modify the specific set of rules or grammars, which is a laborious manual task. The problem of creating

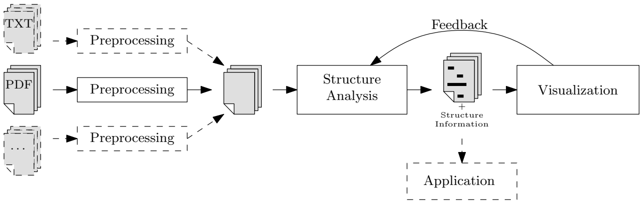


Figure 1: The different components of the system.

a representative set of training and test documents is also not addressed by these methods.

Frameworks for visualizing document structure [14, 15] support basic interactions and editing of meta-data. Both methods show the structure by coloring the background according to the assigned labels. It is possible to correct the labels or to label a document manually, but there is no direct coupling with an underlying structure analysis method which supports the users in these tasks. Thus, the manual corrections are not used to improve the analysis results or reduce the manual efforts for generating reference data.

Semi-automatic methods are used for generating information extraction wrappers for web sites [4, 12]. The wrappers use patterns based on the HTML information in the web sites to extract the requested information. For the generation of the patterns visual interfaces are used that show the web site with the annotated information to a user who can verify and create annotations. The user's input is afterwards used to generate the appropriate patterns for the web site.

3. LOGICAL STRUCTURE ANALYSIS

Our proposed structure analysis system consists of three main components, as shown in Figure 1: Preprocessing, Structure Analysis and Visualization.

In the *Preprocessing* step, the text lines of the documents are extracted and converted into an intermediate representation. During the extraction, only the textual content with its layout and formatting information is preserved. Other content, e.g. images or movies, is discarded. The resulting intermediate representation is independent from the format of the input file.

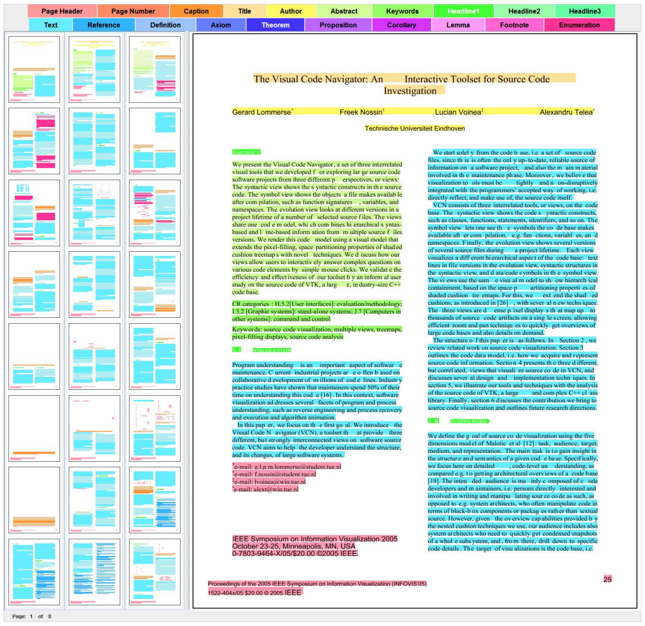
In the *Structure Analysis* step different features are calculated from the layout and formatting properties of the text lines. These features are used by a classifier to analyze the document structure and assign labels to the lines of the document. The features and labels used by the structure analysis are application dependent and different features and labels maybe used for different tasks.

The user is integrated in the Structure Analysis process based on a *Visualization* of the structure analysis results who can directly verify and correct them. The visualization is used during the training phase of the structure analysis in order to reduce the manual effort to create a training set.

3.1 Structure Analysis

During the structure analysis of a document, the whole document is examined line by line. A standard classification algorithm uses a set of layout and formatting features to assign a user-defined label to each line. Three types of layout and formatting features are used in this step:

The first type of features describes the position of a line on the page. The features can be used for identifying header and footer of pages. Lines that appear on top or bottom of a page are headers respectively footers if they have a small font size. In addition to the position of the line on a page, the position within the whole document is regarded as a feature. This is useful to identify labels appearing frequently at a specific region within the document, e.g. titles at the beginning or references at the end.



The Visual Code Navigator: An Interactive Toolset for Source Code Investigation

Gerard Lommerse* Freek Nossin† Lucian Voinea‡ Alexandru Telea*

Technische Universiteit Eindhoven

ABSTRACT

We present the Visual Code Navigator, a set of three interrelated visual tools that we developed f or exploring lar ge source code software projects from three different p erspectives, or views: The syntactic view shows the s yntactic constructs in th e source code. The symbol view shows the objects a file makes availab le after com pilation, such as function signatures , variables, and namespaces. The evolution view looks at different versions in a project lifetime of a number of selected source files. The views share one cod e m odel, whi ch com bines hi erarchical s yntax-based and l ine-based inform ation from m ultiple source f iles versions. We render this code model using a visual model that extends the pixel-filling, space partitioning properties of shad ed cushion treemap s with novel techniques. We d iscuss how our views allow users to interactiv ely answer complex questions on various code elements by simple mouse clicks. We validat e the efficiency and effect iveness of our toolset b y an inform al user study on the source code of VTK, a larg e, in dustry-size C++ code base.

CR categories : H.5.2[User interfaces]: evaluation/methodology; I.3.2 [Graphic systems]: stand-alone systems; J.7 [Computers in other systems]: command and control

Keywords: source code visualization, multiple views, treemaps, pixel-filling displays, source code analysis

1 I NTRODUCTION

Program understanding is an important aspect of softwar e maintenance. C urrent industrial projects ar e o ften b ased on collaborative d evelopment of m illions of cod e lines. Industr y practice studies have shown that maintainers spend 50% of their time on understanding this cod e [16]. In this context, software visualization ad dresses several facets of program and process understanding, such as reverse engineering and process recovery and execution and algorithm animation.

In this pap er, we focus on th e first go al. We introduce the Visual Code N avigator (VCN), a toolset th at provide three different, but strongly interconnected views on software source code. VCN aims to help the developer understand the structure, and its changes, of large software systems.

*e-mail: g.l.p.m.lommerse@student.tue.nl
†e-mail: f.nossin@student.tue.nl
‡e-mail: lvoinea@win.tue.nl
*e-mail: alext@win.tue.nl

IEEE Symposium on Information Visualization 2005
October 23-25, Minneapolis, MN, USA
0-7803-9464-X/05/$20.00 ©2005 IEEE.

We start solel y from the code b ase, i.e. a set of source code files, since th is is often the onl y up-to-date, reliable source of information on a software project, and also the m ain m aterial involved in th e maintenance ph ase. Moreover, we believ e that visualization to ols must be tightly and n on-disruptively integrated with the programmers' accepted way of working, i.e. directly reflect, and make use of, the source code itself.

VCN consists of three interrelated tools, or views, on the code base. The syntactic view shows the code s yntactic constructs, such as classes, functions, statements, identifiers, and so on. The symbol view lets one see th e symbols the co de base makes available aft er com pilation, e.g. fun ctions, variabl es, an d namespaces. F inally, the evolution view shows several versions of several source files during a project lifetime. Each view visualizes a diff erent hi erarchical aspec t of the code base: text lines in file versions in the evolution view, syntactic structures in the syntactic vi ew, and d ata/code s ymbols in th e symbol view. The vi ews use the sam e visu al m odel to sh ow hierarch ical containment, based on the space p artitioning properti es of shaded cushion tre emaps. For this, we ext end the shad ed cushions, as introduced in [26] , with sever al n ew techn iques. The three views are d ense p ixel display s th at map up to thousands of source code artifacts on a sing le s creen, allowing efficient zoom and pan techniqu es to quickly get overviews of large code bases and also details on demand.

The structure o f this pap er is as follows. In Section 2 , we review related work on source code visualization. Section 3 outlines the code data model, i.e. how we acquire and r epresent source code inf ormation. Sectio n 4 presents th e three d ifferent, but correl ated, views that visuali ze source co de in VCN, and discusses sever al design and implementation techn iques. In section 5, we illustrate our tools and techniques with the analysis of the source code of VTK, a large and com plex C++ cl ass library. Finally, section 6 d iscusses the contr ibution we bring to source code visualization and outlines future research directions.

2. R ELATED WORK

We define the g oal of source co de visualization using the five dimensions mod el of Maletic et al [12] : task, audience, target, medium, and representation. The main task is t o gain insight in the structur e an d sem antics of a given cod e ba se. Specif ically, we focus here on detailed , code-level un derstanding, as compared e.g. t o gett ing architectural overv iews of a code base [19]. The inten ded audience is ma inly c omposed of c ode developers and m aintainers, i.e. persons directly interested and involved in writing and manipu lating sour ce co de as such, as opposed to e.g. system architects, who often manipulate code in terms of black-b ox components or packag es rath er than textual source. However, given the ov erview cap abilities provided b y the nested cushion techniques we use, our audience includes also system archit ects who need to quickly get condensed snapshots of a whol e subs ystem, and , fro m there, drill down to specific code details . Th e target of visu alizations is the code base, i.e.

Proceedings of the 2005 IEEE Symposium on Information Visualization (INFOVIS'05)
1522-404x/05 $20.00 © 2005 IEEE

25

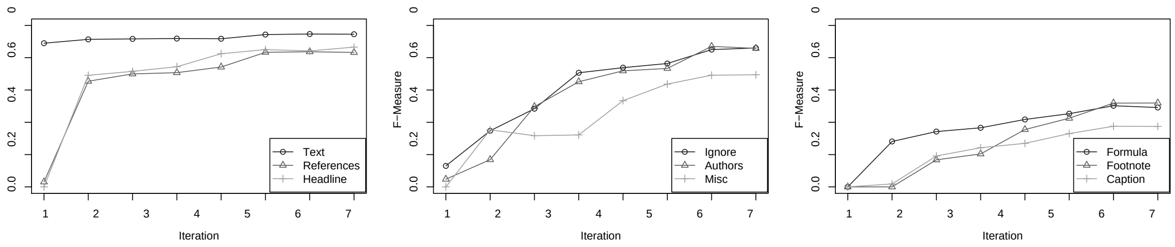
Figure 2: Visualization of the structure analysis results.

The second type of features considers spacing and indention properties. The spacing features describe the distances between a line and the previous one. With this type of features, labels with special spacing properties can be recognized. For instance, the distances between headlines and adjacent lines are usually larger than for normal text lines. Besides the spacing characteristics, the indentions of lines are represented as features. Depending on the type of justification of the text, these features can be used to recognize the beginning and the end of paragraphs. Formulas, captions or larger quotations have usually a different indention than normal text.

The third type of features captures the font style of lines. The font style can be varying by use of different fonts, font size, weights or italic characters. Typically, headlines have a larger font weight and a larger font size than normal text, while headers and footers have usually a smaller font size.

In addition to the formatting and layout features, matches of regular expressions against the line content are used as features as well. They are set to 1 if the regular expressions match, otherwise to 0. With these features it is possible to identify enumerations or captions of figures and tables that start with a common pattern.

The features F_l calculated for a line l are used to build the feature vectors. In order to include some context information in the feature vectors, the feature vector $\vec{f_l}$ that describes line l does not only contain the features calculated for the line itself but also the features for the k previous and k following lines: $\vec{f_l} = (F_{l-k}, \ldots, F_{l-1}, F_l, F_{l+1}, \ldots, F_{l+k})^T$.



(a) (b) (c)

Figure 3: Average F-measure of the structure analysis in the different reference data iterations.

The features of nonexistent lines at the beginning and end of a page are set to zero.

3.2 Visualization

The visualization of the labeled documents allows the user to verify and correct the results of the structure analysis. The visualization consists mainly of two parts: the thumbnails on the left for an efficient navigation in the documents and the detail view in the center for analysis and manual corrections. An example of the visualization is shown in Figure 2.

The thumbnail view shows multiple pages of different documents and is used for efficient navigation in the documents. A thumbnail contains only the bounding boxes of the text lines, which are filled with the color according to the label of the line. The textual content is omitted. The user can open a different page in the detailed view by clicking on the corresponding thumbnail.

The detail view shows one page of the document. The background color of each line is mapped to the label that was assigned during the structure analysis. The user can correct the label of a misclassified line by selecting the line and choosing the correct label from the context menu. The legend for the background color is located in the top row underneath the menu bar.

3.3 Reference Data Creation

Combining the automatic structure analysis with user intervention can be used to adapt the structure analysis to new document types in an efficient way. The required reference data can be generated in the same iterative workflow. In the first iteration no reference data is available to build a structure analysis model, so the user has to label the first document manually. Afterwards, an initial model can be created using the lines in the first document as training set. In the second iteration, a small set of unlabeled documents are chosen and the structure analysis model from the first iteration is applied. Then, the user may correct the automatically generated results. The structure analysis is updated with the corrections made by the user, by adding the corrected lines to the training examples and re-training the machine learning algorithm. This process is continued until the structure analysis reaches a sufficient quality.

4. EVALUATION

The structure analysis approach is evaluated on two different collections of documents. The first collection consists of 250 publications from the proceedings of the computer science conferences IEEE InfoVis 1995-2005, IEEE Vis 1990-2005, SIGMOD 1997-2007, ACM SAC 2005-2008, VLDB 2000-2008 and of articles from INTEGERS Electronic Journal of Combinatorial Number Theory vol. 0-9. The second collection consists of 50 product manuals of different products from various manufactures that are accessible on the Internet. The manuals are collected via a standard search engine using the keyword “manual” and narrow down the results to PDF documents from home pages of consumer electronics manufactures.

4.1 Learning Document Structure

In case of the first collection with the 250 publications, the following semantic labels should be recognized by the structure analysis system: “Title”, “Author”, “Abstract”, “Headline 1”, “Headline 2”, “Headline 3”, “Enumeration”, “Caption”, “Footnote”, “Reference”, “Axiom”, “Definition”, “Lemma”, “Theorem”, “Corollary”, “Proposition”, “Text”. In addition to the formatting and geometry features described in Section 3.1, also regular expressions for matching captions, enumerations, headlines and mathematical components are used here. For the evaluation, the 250 documents are divided into a training collection of 167 documents and a test collection of 83 documents. The documents in the test collection are labeled manually using the tool shown in Figure 2.

At first, the documents in the training collections are labeled according to the method described in section 3.3. In the i-th iteration 2^{i-1} documents are selected randomly from the unlabeled documents in the training collection. To evaluate the efficiency of the training method the intermediate structure analysis of each iteration is evaluated on the test collection. For each label precision P and recall R values are calculated on text lines. In Figure 3 the F-measures $F = 2 \cdot (P \cdot R)/(P + R)$ for different labels in each iteration are shown.

Generally, results in Figure 3 show that three groups of labels can be identified. The first group of labels shown in Figure 3a achieve good results with a few example documents, their F-measure increases to high values during the first three iterations and slowly increases with further training documents in the successive training iterations. The labels shown in Figure 3b benefit most from an increasing number of training documents. Their F-measure increases steadily during all training iterations and achieves good results after the last iteration. The third group of labels shown in Figure 3c cannot be recognized correctly at all with the presented method. Even with an increasing number of training documents these labels do not achieve satisfying recognition results.

Table 1: Performance of different algorithms on INTEGERS articles.

	Title	Author	Abstract	Headline*	Text	Reference	Math. Comp.*	Caption	Enum.	Footnote	*
Nakagawa et al.											
Precision	1.00	0.22	1.00	0.13	0.78	1.00	0.14	1.00		0.00	0.70
Recall	0.46	1.00	0.12	0.24	0.93	0.00	0.02	0.00		0.00	0.20
F-Measure	0.63	0.36	0.21	0.17	0.85	0.00	0.04	0.00		0.00	0.31
Ratté et al.											
Precision	0.83			0.24					1.0		
Recall	0.14			0.24					0.0		
F-Measure	0.24			0.24					0.0		
Proposed System											
Precision	1.00	0.67	0.76	0.60	0.93	0.93	0.83	1.00	0.00	1.00	0.97
Recall	1.00	0.22	0.79	0.51	0.97	0.91	0.81	0.00	1.00	0.00	0.93
F-Measure	1.00	0.33	0.78	0.55	0.95	0.92	0.82	0.00	0.00	0.00	0.95

Table 2: Performance of different algorithms on computer science publications.

	Title	Author	Abstract	Headline*	Text	Reference	Math. Comp.*	Caption	Enum.	Footnote	*
Nakagawa et al.											
Precision	1.00	0.46	0.89	0.20	0.72	1.00	1.00	1.00		0.08	0.24
Recall	0.38	0.21	0.73	0.02	0.99	0.00	1.00	0.00		0.17	0.12
F-Measure	0.56	0.29	0.80	0.03	0.83	0.00	1.00	0.00		0.11	0.16
Ratté et al.											
Precision	1.00			0.92					0.21		
Recall	0.54			0.81					0.47		
F-Measure	0.70			0.86					0.29		
Proposed System											
Precision	0.88	0.63	0.47	0.77	0.94	0.97	0.00	0.95	0.25	1.00	0.50
Recall	1.00	0.92	0.43	0.82	0.96	0.96	1.00	0.87	0.15	0.00	0.46
F-Measure	0.93	0.75	0.45	0.80	0.95	0.97	0.00	0.91	0.19	0.00	0.48

Table 3: Accuracy values based on the performances shown in Table 1 and Table 2.

	INTEGERS	Computer Science
Nakagawa et al.	0.73	0.71
Ratté et al.	0.14	0.69
Proposed System	0.91	0.91

The need for manual interaction is reduced by using the structure analysis during the reference data creation. Within the first two or three iterations, the automatic method recognizes already the majority of text lines correctly. Only the labels of miss-classified lines must be corrected manually be the user.

4.2 Use Case: Publications

With the labeled training collection a new structure analysis is trained and compared to the methods of Nakagawa et al. [10] and Ratté et al. [13]. Nakagawa et al. described an algorithm for extracting structure information and mathematical components from publications. The method of Ratté et al. is a graph based method that uses linguistic information to identify titles, headlines and enumerations in documents. For all methods, the precision, recall and F-measure for each label available in the reference data are calculated on text lines. The results on the INTEGERS articles are shown in Table 1 and Table 2 shows the results on the computer science publications.

Summing up, in Table 3 the accuracy of the different methods from the INTEGERS articles and the computer science publications are shown. From the results in Table 1, Table 2 and Table 3 it is evident that the performance of the algorithms depends on the type of the document collection. The algorithm of Nakagawa et al. performs almost equally on the INTEGERS articles as well as on the computer science publications. In particular, the system of Ratté et al. achieves much higher accuracy on the computer science publications than on the INTEGERS articles. The proposed system outperforms the two others, on both, the INTEGERS articles and the computer science publications.

Comparing the results in Table 1 with Table 2, it is evident that predefined structure analysis algorithms have the drawback to work only for a specific document collection. Adaptations of these algorithms to different document types results in designing and implementing additional rules or grammars. In contrast, the machine learning approach of the proposed system can easily be adapted to different document collections and achieves very high recognition rates. Basically, only the feature set used for the structure analysis has to be adapted to the specific properties of the new document collection.

4.3 Use Case: Product Manuals

As already mentioned, in addition to the computer science publications and INTEGERS articles, the proposed system is easily adapted to process a collection of product manuals. In this third type of documents, the following structural labels should be recognized: “Title”, “Headline”, “Table of Content” (TOC), “Hint” and “Text”. Here, a new feature set with geometry and formatting features is implemented. A

Table 4: Performance of the proposed system on product manuals.

	Title	Headline	Text	TOC	Hint
Proposed System					
Precision	1.00	0.93	0.95	0.83	0.46
Recall	0.14	0.80	0.96	0.81	0.14
F-Measure	0.25	0.86	0.95	0.82	0.21

regular expression scheme is used to match headlines. The proposed system is tested with 10-fold cross-validation on these new settings. The results are shown in Table 4.

It is evident that the system is able to identify text, headlines and TOC within the product manuals. Differences in the layout of front pages yield into complicated recognition of titles. Hints often occur in manuals and are highlighted by different background color or have a border. The background colors as well as the borders are removed during the reading of the PDF document; therefore this information cannot be used for structure analysis. In contrast, headlines and TOC have special geometric characteristics and are contained in almost all manuals.

Summarizing Table 4, the proposed system achieved a promising accuracy of 0.94. It is shown that the proposed system for scientific publications can easily be extended to the requirements of the structure recognition for product manuals.

5. CONCLUSION AND FUTURE WORK

Existing structure analysis methods are always designed for dedicated document collections. Their adaptation to new document types requires an expensive manual re-implementation. With the proposed machine learning approach, such adaptation efforts can significantly be reduced. As Section 4.2 and Section 4.3 show, the application of the proposed system to three different document collections shows by the majority much better accuracy values than two state-of-the-art methods.

The suggested coupling of machine learning with interactive visualization techniques reduces manual efforts in creating reference data very clearly. Section 4.1 explains how the need for manual user interaction can considerably be decreased by integrating the user in the manual verification and improvement of automatically derived classification results.

For further improvements of the proposed system new methods for learning regular expressions and keywords from example documents will be integrated. Thereby, simple regular expressions could be learned automatically which would reduce manual interaction efforts much further. In addition to current efforts in developing new OCR technology for the retroconversion of historical documents, the proposed structure analysis framework shall be extended for processing structure information captured from raster images instead of PDF files.

6. ACKNOWLEDGMENT

The work presented here was partially funded by the German Federal Ministry of Economy and Technology (BMWi) under the THESEUS project and by the German Research Society (DFG) under the grant GK-1042, Explorative Analysis and Visualization of Large Information Spaces, Konstanz.

7. REFERENCES

[1] A. Anjewierden. AIDAS: Incremental Logical Structure Discovery in PDF Documents. In *ICDAR '01: Proceedings of the Sixth International Conference on Document Analysis and Recognition*, 2001.

[2] R. Brugger, A. Zramdini, and R. Ingold. Modeling Documents for Structure Recognition Using Generalized N-Grams. In *ICDAR '97: Proceedings of the 4th International Conference on Document Analysis and Recognition*, 1997.

[3] J. C. Handley, A. M. Namboodiri, and R. Zanibbi. Document Understanding System Using Stochastic Context-Free Grammars. In *ICDAR '05: Proceedings of the Eighth International Conference on Document Analysis and Recognition*, 2005.

[4] U. Irmak and T. Suel. Interactive wrapper generation with minimal user effort. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, New York, NY, USA, 2006.

[5] Y. Ishitani. Logical Structure Analysis of Document Images Based on Emergent Computation. *IEICE - Trans. Inf. Syst.*, E88-D(8), 2005.

[6] J. Kim, D. X. Le, and G. R. Thoma. Automated labeling in document images. In *Proc. SPIE: Document Recognition and Retrieval VIII*, 2001.

[7] S. Klink, A. Dengel, and T. Kieninger. Document Structure Analysis Based on Layout and Textual Features. In *DAS '00: Proceedings of the 4th IAPR International Workshop on Document Analysis Systems*, 2000.

[8] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[9] G. Nagy. Twenty Years of Document Image Analysis in PAMI. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(1), 2000.

[10] K. Nakagawa, A. Nomura, and M. Suzuki. Extraction of Logical Structure from Articles in Mathematics. In *Proceedings of Mathematical Knowledge Management, Third International Conference*, volume 3119 of *Lecture Notes in Computer Science*. Springer, 2004.

[11] A. M. Namboodiri and A. Jain. *Document Structure and Layout Analysis*. Advances in Pattern Recognition. Springer-Verlag, London, 2007.

[12] J. Raposo, A. Pan, M. Álvarez, J. Hidalgo, and A. Viña. The wargo system: Semi-automatic wrapper generation in presence of complex data access modes. *International Workshop on Database and Expert Systems Applications*, 2002.

[13] S. Ratté, W. Njomgue, and P.-A. Ménard. Highlighting Document's Structure. In *Proceedings of World Academy of Science, Engineering and Technology*, volume 25, 2007.

[14] M. Rigamonti, O. Hitz, and R. Ingold. A framework for cooperative and interactive analysis of technical documents. In *GREC '03: Fifth IAPR International Workshop on Graphics Recognition*, 2003.

[15] S. Yacoub, V. Saxena, and S. N. Sami. Perfectdoc: A ground truthing environment for complex documents. *International Conference on Document Analysis and Recognition*, 0, 2005.