# Analysis of Documents Born Digital

23

Jianying Hu and Ying Liu

## Contents

J. Hu (✉)
IBM T.J. Watson Research Center, Yorktown Heights, NY, USA
e-mail: jyhu@us.ibm.com

Y. Liu
Korea Advanced Institute of Science and Technology (KAIST), Yuseong-gu, Daejeon, Republic of Korea
e-mail: yingliu@kaist.ac.kr

D. Doermann, K. Tombre (eds.), *Handbook of Document Image Processing and Recognition*, DOI 10.1007/978-0-85729-859-1_26,
© Springer-Verlag London 2014

**Abstract**

While traditional document analysis has focused on printed media, an increasingly large portion of the documents today are generated in digital form from the start. Such "documents born digital" range from plain text documents such as emails to more sophisticated forms such as PDF documents and Web documents. On the one hand, the existence of the digital encoding of documents eliminates the need for scanning, image processing, and character recognition in most situations (a notable exception being the prevalent use of text embedded in images for Web documents, as elaborated upon in section "Analysis of Text in Web Images"). On the other hand, many higher-level processing tasks remain due to the fact that the design purpose of almost existing digital document encoding systems (i.e., HTML, PDF) is for display or printing for human consumption, not for machine-level information exchange and extraction. As such, significant amount of processing is still required for automatic information extraction, indexing, and content repurposing from such documents, and many challenges exist in this process. This chapter describes in detail the key technologies for processing documents born digital, with a focus on PDF and Web document processing.

**Keywords**

Content extraction • Digital libraries • Digital publishing • Document layout analysis • Document object model • Document understanding • Page segmentation • Portal document • Web documents

## Introduction

The work on analysis of document born digital started in the mid- to late 1990s, on PDF documents, plain text documents such as email, and analysis of Web documents as semi-structured data [33]. The earliest book of collections of work on Web document analysis appeared in 2004 [4]. Because of the degree of complexity involved in PDF and Web documents and the phenomenal growth in prevalence of these two encoding systems as popular digital publishing media, significant amount of work has been carried out in these two areas and tremendous progress has been made in the past 15 years. This chapter thus focuses on these two dominant areas: PDF Document Process and Web Document Process. For each area, detailed explanations are given on the motivation and research challenges, the key approaches, and state of the art for addressing the challenges, as well as promising new directions. Also provided are descriptions of available software and tools and a comprehensive list of references for further reading.

## Challenges and Recent Advances

### PDF Document Analysis

Along with the rapid expansion of digital libraries, PDF documents have become an important data source for many applications, such as information retrieval or data integration. The demand for extracting and analyzing information from PDF documents is increasing. In order to unlock the information in PDF documents, one needs to not only understand the PDF file format and syntax at object, file as well as document level, but also become familiar with representative PDF document analysis tools and methodologies.

### Motivation

After years of massive digitization, multiple applications and file formats are launched as "digital paper" or "portable documents." The representative examples are Novell's *Envoy*, Common Ground's *Common Ground*, and Adobe's *Acrobat*. All of them attempt to provide a platform- and application-independent way of distributing electronically produced final form documents. Acrobat and its underlying Portable Document Format (PDF) are winning the race to becoming a de facto standard. Version 1.0 of PDF was published in 1993, and PDF has been gradually established as the most popular format of digital document distribution. As a result, the analysis and understanding of PDF documents has become a crucial and vital step for all the further applications such as information extraction.

### Challenges

Although PDF has become the defacto standard for the distribution of electronic documents, especially in digital libraries and the World Wide Web, the print-oriented nature of PDF leads to serious drawbacks as follows:

- Little structural information
  Since PDF was originally designed for the final presentation of a document, most PDF documents are untagged. They contain very little or no explicit structural information about the contents such as words, text lines, paragraphs, logos, and figure illustrations, which makes content extracting or repurposing a challenging task. Particularly, it is difficult to reuse, edit, or modify the layout or the content of the document.
- Render ordering problem
  The main advantage of PDF documents is to preserve the original document look. As a document reviewer, one cares about the final PDF layout and content, instead of the process when the page is produced. Many different PDF programs (e.g., PostScript) can generate the same document appearances and rendered

effect on the screen; however, they may follow different render ordering. A given PDF file might be structured to write out text a word/character at a time, or render a specific type of texts (e.g., titles) first or some other algorithm that tries to optimize on-screen rendering. Very often, the render ordering is totally different from the "reading order." For multicolumn text, the file is sometimes rendered across the columns by hopping across the inter-column gutter. Although the lack of standard render sequence does not affect the PDF document displaying and reading, it heavily impacts the performance of document structure analysis and understanding.

- Object overlapping problem
  If a PDF document has rich graphical contents, there is a high probability to have block overlay phenomena, e.g., a textual company name and a company logo are embedded in an image, which makes the analysis more complicated. Moreover, even if it is possible to identify all the page objects in a PDF document, they are far from reflecting the logical structure or logical components of document. For example, a text object may only have part of the characters of a word. Path objects are often just a fraction of the whole figure, e.g., one line in a line chart. Such overlapped objects introduce additional burdens and chances in PDF understanding and logical structure analysis.
- Diverse document layouts and inconsistent fonts
  Because PDF documents are generated by different approaches (see details in section "Analysis of Text in Web Images") and widely used in many domains from scientific papers to scanned transcripts and from advertising flyers to statistical reports. The diverse document layouts and complicated typesetting make PDF documents prone to errors, particularly in multicolumn documents with inconsistent text font types or when the original document has been created at 90° rotation or inter-column gutter is very narrow.
- Object extraction
  Although extracting and analyzing the whole PDF file is a popular research topic, sometimes one needs to be able to extract parts of the whole file and use it in Web pages, word processing documents, PowerPoint presentations, or desktop publishing software. Such document segments include not only paragraphs or sentences in specific locations but also typical logical components such as tables or images. Fortunately there are many systems/tools that can satisfy this requirement. For example, the full version of Adobe Acrobat can extract individual images or all images from a PDF and export in various formats such as EPS, JPG, and TIFF. In addition, many third-party PDF extraction Software Tools exist. However, these tools often cannot fulfil the task of object extraction fully automatically. Instead, human involvement such as manually selecting the object boundary in document is needed.

## Overview of PDF Models and Objects

### Outside PDF

Huge number of documents are either created or converted into PDF format from other document formats. They can be generated from nearly all standard

**Table 23.1** The representative PDF generation methods

| PDF generation methods | Details |
|---|---|
| Method 1 | Create PDF directly by PDFWriter from document authoring systems |
| Method 2 | Create PDF from PostScript files by Adobe Distiller software |
| Method 3 | Create PDF by Adobe Acrobat Capture based on a bitmap image document in TIFF form |

applications (Microsoft Word, Microsoft Excel, Microsoft PowerPoint, Adobe Photoshop, Adobe InDesign, Adobe Illustrator, Quark XPress, etc.) by a pseudo printer driver called *PDFWriter* or directly from PostScript using Adobe's *Distiller*, which is an interpreter to convert PostScript files to Adobe PDF. The generation methods of PDF documents can be classified into the following types shown in Table 23.1.

### Inside PDF

PDF files consist of a number of distinct *indirect objects* which may be distributed randomly within the file. These objects include text objects, image objects, and path objects. Every PDF document has a root object (catalog) containing indirect references to various structures. For faster access in viewer software, PDF pages are arranged in a tree structure, instead of linearly. The page tree is a balanced tree of page objects, with page objects at the leaves, and allows faster access to documents containing multiple pages. Please see PDF file structure in Fig. 23.1. Please note that page object does not reflect to the logical structure in a PDF document. Similarly, a text object may only contain part of a word (e.g., several characters) and the path object contains part of graphical illustrations (e.g., a line in a figure). Based on the nature of PDF document content stream and the frequently happened block overlay among different objects, how to analyze and understand the logical component of a PDF document is a challenging problem.

As shown in Fig. 23.1, images are represented as separate child objects in the pages tree model. This makes them easy to identify and no further processing needs to be done to segment or classify these elements. The thumbnail entry in a page object is a tiny JPEG-compressed image of the page, which can be used for navigation in a viewer. The Annotations contains several standard "hyper-features" features, such as link bookmarks and "yellow sticks." New features can be added to Acrobat software by third-party plug-ins written using the Acrobat SDK.

## PDF Document Understanding Approaches

Because the analysis of document images has a longer history than that of PDF documents, usually there are two ways to analyze PDF documents: either directly analyze the PDF documents or convert the PDF documents into bitmap images and perform existing document image segmentation techniques.

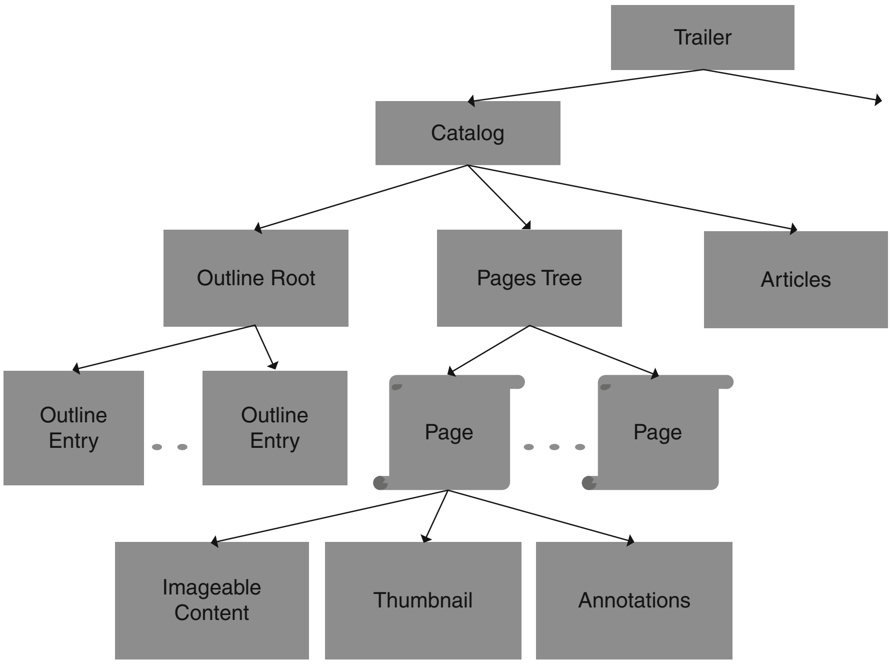

**Fig. 23.1** Part of the main PDF file structure

## Image-Based Approaches

Since the middle of last century, image documents are the main pre-Web documents, which are dominant in the document understanding and analysis field. Image documents are scanned from pseudo binary hardcopy paper manuscripts with a flatbed sheet-fed or mounted imaging devices. Extensive works have been done in detecting logical structure in scanned bitmap image documents for document understanding and information extraction [37]. Although most methods are not originally designed for PDF document analysis, many of them can also be reapplied to PDF documents. They simply make use of a bitmap rendition of each PDF page and apply methods similar to those designed for scanned pages. The typical preprocessing step is to render a PDF document into a bitmap. Such bitmap images will be accepted as inputs, and then text, graphics, and images are identified and extracted. Because of the limited information available in the bitmap images, most studies are limited to business letters, technical journals, and newspapers, where object overlaps are minimal.

In order to identify the textual information as well as understand the document semantically, Optical Character Recognition (OCR) [30] and many other techniques are necessary. This processing also requires the recognition of fonts, as well as words and characters. The resulting PDF is usually much more compact than the TIFF image and can be searched for arbitrary phrases and keywords by all of the standard PDF search engines.

**Table 23.2** Representative literatures on PDF document analysis

| Literatures | Methods/tools | Contributions |
|---|---|---|
| Smith and Brailsford [40] | Segment a PDF document page into different image and text blocks | Covert a PDF document into a pool of logical, typed blocks |
| Anjewierde [3] | AIDAS: converted a PDF to a set of text, graphics, and image objects | Incremental Logical Structure Discovery in PDF documents |
| Futrelle et al. [15] | Obtain the graphics primitives in object form | Recognizing graphics from PDF |
| Hadjar et al. [18] | Xed: applying TIFF image layout analysis methods on PDF documents | Extracting hidden structures from PDF documents using text and graphic objects |
| Chao and Fan [8] | Separated each PDF page into three layers: text, image, and vector graphics layer | Identifying logical components in a PDF page |
| Marinai [35] | Greenstone package | Extract basic metadata from PDF documents |
| Hassan [19] | PDF parse method based on *PDFBox* | Object-level document analysis of PDF files |

### Direct-Processing Approaches

PDF document content stream lists all the page objects, such as text, image, and path. Thus another way to discover the logical components of a document is to analyze and interpret the layouts and attributes of the page objects so as to correctly break or group them into different logical components.

With the help of many commercial or open-source tools such as XPDF (http://www.foolabs.com/xpdf) or PDFBOX (http://pdfbox.apache.org/), the page objects can be extracted from PDF sources directly. The major work is to segment the texts in a PDF page into regions containing texts with similar attributes. The two most important attributes are point size and font name, which are stored as part of the page resources in each node of the PDF page tree in Fig. 23.1. These valuable attributes are used as key features with predefined heuristic-based methods for tagging the various segments.

### The State of the Art

Based on the fundamental approaches introduced above, several representative works are selected and listed chronologically in Table 23.2, which not only provides summaries and comparisons of these methods but also reflects the research progress and trend in this area.

Similar to other document media, there are three approaches to analyze the PDF document structure: model-driven (top-down), data-driven (bottom-up), or hybrid approach. Hybrid algorithms can be regarded as a mix of the above two approaches.

- Model-driven approach is also called top-down approach. If PDF document structure and the page model are known in advance, the top-down approach can be used based on the predefined page templates. To apply this method, one needs to define the PDF document model or understand the page template in advance. Top-down approach starts from the whole PDF document page and iteratively split it into smaller geometric elements based on paragraph breaks or intercolumn gutters. The splitting procedure stops when some criterion is met and the ranges obtained at that stage constitute the final segmentation results. Many existing top-down methods in image document analysis field can be reused.
- If the layout information is unknown in advance, a bottom-up method is adopted. Bottom-up algorithms usually start from the lowest-level document objects (e.g., the characters or image pixels) and cluster them into larger connected components such as words, which are then be grouped into lines or zones. Plain text document analysis methods, such as column/layout detection or space analysis, can be reused here.

Although researchers keep proposing new methods/systems to improve the performance of PDF document understanding, most of them obtain the low-level document objects by taking advantage of text extraction tools, such as open-source libraries for processing PDFs: XPDF library, the Apache PDFBox$^{TM}$ library (http://pdfbox.apache.org), PDFlib Text Extraction Toolkit (TET http://www.pdflib.com/products/tet), PDF2Text (www.pdf2text.com), etc. Then diverse techniques are adopted to have a better understanding on the document content and structure. Most literatures in PDF understanding field adopt bottom-up methods that share the following main steps:

- Step 1: Extracting diverse document objects and low-level data in PDF
  Usually different PDF extraction tools are used to extract the objects and low-level data. JPedal is a Java package that can extract texts, images, annotations, and forms from PDF. The Apache PDFBox$^{TM}$ library is an open-source Java tool to extract texts from PDF. Smith and Brailsford [40] seem published the first paper that deals with the analysis of PDF files. They introduced a new definition of an Encapsulated PDF (*EPDF*), which segments a PDF document page into different image and text blocks. The Adobe Acrobat SDM is used to obtain textual objects.
  In order to minimize the potential logical components overlay and the interference between different logical components in PDF documents, researchers separated a PDF page into three layers: text, image, and vector graphics layer. Anjewierde [3] extracted the logical document structure by converting a PDF document into a set of text, graphics, and image objects using XPDF library. Ten different classes of layout objects (text, line, rectangle, images, curve, etc.) are extracted and each class has ten features (position, font size, color, line width, etc.).
  Chao and Fan [8] saved each layer as an individual PDF document. Based on the type of each layer, they decided whether to analyze the PDF document directly or convert it into bitmap images then perform image analysis methods. Text and image objects are obtained directly from the PDF sources. They extracted the

font, size, line spacing, and color as the text style as well as the text strings as the content. Shapes of the images and the masks are extracted for image components. Lines and vector objects are obtained from a bitmap image.

Hadjar et al. [18] also proposed a new tool called Xed for extracting hidden structures from PDF documents using text and graphic objects. Xed extracts all the objects from a PDF document including text, images, and graphics. They merge the typical low-level extraction methods applied on PDF files with the traditional layout analysis performed on TIFF images.

Futrelle et al. [15] adopted the *Etymon PJ Tools* (http://www.etymon.com/epub.html) library to obtain the graphics primitives in object form from PDF documents. The extracted information is at a much finer granular level than what is required for document analysis.

Hassan [19] implemented his PDF parse method based on *PDFBox*, an open-source library for processing PDFs. They used *PDFStreamEngine* class to extract and store the objects from a page. Besides, this paper explains how the relevant PDF instructions can be processed to form a set of objects suitable for document analysis.

Marinai [35] designed the Greenstone package, a full integration with multiple popular digital library software, to extract basic metadata from PDF documents. The metadataPDFPlug is a plug-in, which imports PDF documents based on PDF reader XPDF and the Ghostscript libraries.

- Step 2: Forming words from characters, lines from words, and layout structures from lines by discriminating line spacing, indentation, and basic geometric relationships

  Almost all the related works started with the low-level objects extracted in Step 1 and tried to convert the textual pieces into larger textual elements in a bottom-up way: from characters to words, lines, paragraphs, and even page columns. Usually the word-finding algorithm of the API within the Acrobat Exchange viewer can identify the logical words. Based on the geometric information of each word, line, or paragraph, even larger textual regions can be generated by combining the font name and font size and comparing the start, middle, and end point among adjacent textual elements. The detailed steps can be found in [40] Besides font name and point size features, they also considered a set of metrics holding the important geometric features of the font: x-height, caps height, serif, sans serif, stem width, and so on. Similarly, Chao and Fan [8] used an attribute vector, which included font name, font size, color, and orientation as well as the four corners of the quad(s) provided by Adobe's Acrobat word finder to represent the style of the words. Grouping adjacent textual pieces step is same as other papers. Marinai [35] adopted *Pdftohtml* to merge text lines into paragraphs by preserving the texts in reading order. Coordinate information is analyzed for each textual piece and merges together textual blocks in horizontal and vertical directions. Hassan [19] proposed a best-first clustering algorithm, a robust bottom-up segmentation algorithm based on visual principles even on complex layouts such as newsprint. There is no doubt that the performance of layout structure analysis is improved, but complex layout structure still needs human intervention.

- Step 3: Identifying document logical components
  Usually the logical components are identified based on the structure components extracted in previous steps. Various approaches were proposed and implemented. Top-down combination method is used when the document-style information (e.g., font parameters, size of section titles, line spaces) is known. Another popular method is discovering a hierarchy of layout objects then mapping the layout objects to the logical components. Anjewierde [3] discovered the logical structures based on further grammar processing by processing text, image, and vector graphics layer separately to avoid the effect of the object overlapping. Smith and Brailsford [40] used PDF to represent block-based documents, by carrying each block with the resource requirements as well as geometrical and logical information. With his extension proposal, the authors converted a PDF document into a pool of logical typed blocks. A block formatter called Juggler allows these blocks to be extracted and reused. In order to automatically extract administrative metadata, the authors in [35] developed a novel package *pdf2gsdl* to extract PDF document metadata. They used a Multi-Layer Perceptron (MLP) classifier and the neural classifier to identify regions and label metadata to each of them.

## Specific Object Recognition in PDF

Instead of extracting all the contents in a PDF document, one may be interested in a part of the contents or specific document objects. The lower-level objects include symbols, words, sentences, and paragraphs, which can be easily parsed and identified. The higher-level objects refer to textual granularities that need further analysis. Table is one of representative examples.

In the last 20 years, over 200 papers have been published on table recognition field. Zanibb et al. [47] provides a survey with detailed description of existing approaches and systems. In addition, a detailed bibliography on table segmentation and extraction can be found in http://www.visionbib.com/bibliography/char969.htm. To extract data from tables, two separate tasks need to be fulfilled:

- Step 1: Detecting the table boundary from the document
- Step 2: Extracting the table cells as well as other table-related information

Researchers in the automatic table extraction field largely focus on analyzing the table in a specific document media. Most works focus on three main document types: image-oriented, HTML-oriented, and the plain text-oriented. Although many approaches and systems are available for these document types, few methods are designed to deal with PDF tables.

Different media requires different table boundary detection and table decomposition methods. Although they are not new research topics, very few published papers provided detailed algorithms to extract tables from PDF documents. Most PDF table extraction works have similar preprocessing steps:

- Sorting texts by the coordinates to avoid the order uncertainty brought by the document creator (detect the table boundary first and then only sort the small areas)

- Merging text elements into lines, combing lines into blocks
- Analyzing each line/block to decide whether it is belonging to a table boundary

There are several representative works on table detection in PDF documents. Wasserman et al. [44] designed an algorithm to detect potential table regions in a PDF document and to analyze the table structure. First, they converted 100 scientific documents in PDF format into TIFF image format, then treated a set of word boxes as the input of the algorithm, and used the word segmentation technique in [43]. They yielded a set of potential table-region bounding boxes through the processing of a set of modules: word-box consolidation and text line elimination module, vertical-range module, horizontal-range module, and table-region extraction module. The main shortcoming of this paper is the lack of the details. Although they realized the importance of the precise definition of table, there is no definition for all the terms. In addition, there is no experiment and evaluation.

pdf2table [46] is an early-stage tool to extract tables from PDF files. Instead of the typical textual information extraction tools such as PDF2TET or TET, pdf2table uses the pdf2html (http://pdftohtml.sourceforge.net) developed by Gueorgui Ovtcharov and Rainer Dorsch to get all text elements (i.e., strings) with the absolute coordinates in the original file. All the returned texts are saved in a structured data format (XML file). Heuristic-based approach is also used to detect and decompose table boundary. However, the performance is far from unsatisfying. To remedy the deficiency of the tool, the authors implemented a graphical user interface, which enables the user to make adjustments on the extracted data.

Tamir Hassan and Robert Baumgratner [20] also used PDFBox to extract texts from PDF files and try to recognize tables. They grouped tables into three categories based on the existing of horizontal and vertical ruling lines. However, ruling lines are not required in real situations. AIDAS system, which extracted the logical document structure from PDF documents [3], used shallow grammars to detect the logical elements as the authors admitted, and no result and evaluation is presented.

Liu et al. designed TableSeer, a table search engine based on extracting table metadata from PDF documents. The early version of TableSeer [31] decided the table boundary within a PDF page by classifying a document page into a set of boxes using a top-down method. Then each box is analyzed and classified into table candidate box or not. The later version of TableSeer adopted a bottom-up method by labeling each document line with the table candidate boxes as table line or non-table line. Both heuristic methods and machine learning-based methods are used in this work.

PDF-TREX [38] recognized each PDF table as a 2-dimensional grid on a Cartesian plane by analyzing a set of basic content elements heuristically in a bottom-up way.

PDF table extraction is a challenging problem because of many reasons. First, double-column even three- or four-column PDF documents introduce more difficulty in data extraction. In addition, many methods are only good for the lucid tables. Second, many heuristics cannot be easily extended and reapplied to other datasets. Each of them will incur a lot of noisy results. Third, widely used thresholds are very important to the final performance and it is impossible to deal with all the cases with a single fixed threshold value.

## Web Document Analysis

Over the last 20 years, the Web has rapidly become a dominant media for information exchange, where users ranging from professional publishing institutions such as New York Times to individual bloggers routinely publish and access documents on demand. The availability and extent of this vast information source coupled with the decentralized nature of the medium pose both increasing demand and significant challenges for automated content analysis. Such content analysis is crucial for applications such as content repurposing, information extraction, Web mining, and Web security. This section presents an overview of the four key challenges in this area: Web Page Segmentation, Web Content Extraction, Analysis of Text in Web Images, and Web Document Modeling.

## Web Page Segmentation

The Web as a publishing medium has become significantly more mature and more sophisticated over the last decade, while at the same time more decentralized. As a result, Web page layout has become complex and diverse. The segmentation of a Web page into visually and semantically coherent components has emerged as one of the most important Web document analysis tasks supporting many important applications and serving as the preprocessing step for comprehensive Web content extraction.

### Motivation

Web page segmentation is the crucial first step in identifying different types of information (i.e., heading, navigation bar, central content), and the informative and non-informative portions of the Web page. Such information is very useful in Web ranking and Web data mining applications, as well as Web mining tasks such as Web document clustering and duplicate detection.

The identification of segments also plays an important role in displaying Web pages on-screen-space-constrained devices such as smartphones and PDAs. While these mobile devices are being increasingly used to access the Web, the small size of the device screens often limits the user experience, as the vast majority of the Web sites are designed for viewing on a standard desktop or laptop screen. Proper page segmentation is vital in providing the ability to automatically adapting such sites for effective viewing on small screens.

### Approaches

Since most Web pages can be parsed into a DOM (Document Object Model) using a standard HTML parser, and by definition the DOM tree contains all element needed to render a Web page, it is a natural place to start with any Web page segmentation task. Given the DOM tree, the problem of Web page segmentation becomes one of the grouping DOM elements into coherent regions. A variety of methods have been developed to address this problem. These methods can be roughly categorized into the following three categories based on the specific grouping mechanism.

## DOM Tree Centric Approach

A large number of Web pages follow a common template at the highest level, which includes header, footer, left side bar, right side bar, and body. These high-level content blocks typically follow certain layout conventions which can be used to derive rules for their classification using information coded in the DOM tree. Since each high-level content block (especially the body block) can still be heterogeneous, more rules based on HTML tags (explicit separators) and layout characteristics (implicit separators) need to be learned for potential further segmentation. The approach involves the following two steps [10]:

- Step 1. High-level content block detection
  The DOM tree, which contains the position and dimension information for each node, is traversed to classify each node as belonging to the one of the five high-level blocks. The classification is based on shape and location conventions typically followed by each of these blocks. For example, the header tends to be located at the top portion of the page and also tends to have a "flat" shape, i.e., with low height to width ratio. Instead of devising handcraft rules on spatial location and aspect ratios, a more systematic approach is to extract these spatial features (top, bottom, left, right, and aspect ratios) for each element from the HTML DOM tree and then train a classifier such as nonlinear support vector machine using a set of labeled results.
- Step 2. Further segmentation of content body
  To further identify finer grouping within each content body, first a set of explicit separators are identified. Explicit separators are HTML elements that can be determined to be a separator based on its own properties. They include elements with tags, such as <p>, <hr>, <table>, <td>, <ul>, <h1>, and <div>, and thin images (which can be readily detected by examining the aspect ratio and can be easily identify with one tree traversal).
  While these explicit separators are easy to extract from the DOM tree, they don't necessarily cover all potential separators, since certain separation is visual and not directly coded in the DOM tree. In order to identify these separators, one needs to first project the basic content blocks along the horizontal and vertical axes to generate the project diagrams, and then identify gaps along on 2 each axis as implicit separators. This is very similar to the projection method used in scanned document image segmentation (▶Chap. 5 (Page Segmentation Techniques in Document Analysis)).
  A natural extension to the projection based approach for page segmentation is the X-Y cut algorithm that has found great success in scanned document image segmentation. Like in the case of scanned documents, repeated cuts can be made along the X and Y direction based on a coherence measure, until desired number of blocks or a predefined coherence threshold is reached.

### Vision-Based Content Structure Approach

The projection-based method and X-Y cut-based algorithms are attempts to combine DOM structure with visual cues for page segmentation. Another approach that takes this one step further and puts more emphasis on visual representation is the algorithm called VIPS (Vision-Based Page Segmentation) [5].

In the VIPS algorithm, instead of operating solely on the DOM tree, a vision-based content structure of a page is deduced by combining the DOM structure and the visual cues. From the root node of the DOM tree, a visual block extraction process is applied to extract visual blocks at each level based on rules designed to capture four visual cues:

- Tag cue: A node is divided if it contains one of the explicit separators as defined in the previous section.
- Color cue: A node is divided if its background color is different from one of its child.
- Text cue: A node is not divided if most of the children of a DOM node are Text nodes.
- A DOM node is divided if the standard deviation of size of its children is larger than a threshold.

All extracted blocks are then placed into a pool for visual separator detection. Visual separators are defined as horizontal or vertical gaps that do not intersect with any block. The algorithm for visual separator detection proceeds in an iterative manner by traversing through all the blocks in the pool. Each separator is then given weights based on the following set of heuristics:

- The larger the distance between blocks on different side of the separator, the higher the weight.
- The weight of a separator is increased if:
  - A visual separator is at the same position as an explicit separator such as <HR>.
  - The differences of font properties such as font size and font weight are more pronounced on two sides of the separator.
  - Background colors are different on two sides of the separator.
- When the structures of the blocks on opposite sides of the separator are similar (e.g., both are text), the weight of the separator is decreased.

Finally, the visual structure construction process starts from the separators with the lowest weight, and the blocks opposite these separators are merged to form new virtual blocks. This process iterates until a predefined level of granularity or degree of coherence is reached.

### Graph Theoretic Approach

The methods described in the previous two sections rely heavily on heuristics applied to tree-based greedy search, which tend to lead to local minima that are not necessarily optimal. To address this problem, Chakrabarti et al. [7] developed an approach which is based on a weighted graph constructed over nodes of the DOM tree, where the cost of assigning pairs of DOM elements to different of identical segments can be explicitly coded. This construct then allows the definition of a global cost of a segmentation, which can be minimized in a disciplined manner using well-established graph clustering algorithms.

The approach is based on the following formulation. Given a DOM tree, let $N$ be the set of nodes. A graph is constructed whose node set is $N$ and whose edges have a weight that represents a cost of placing the adjacent nodes in different segments. $D_p : L \to R^{\geq 0}$ represent the cost of assigning a specific label to the node $p$. Let $V_{pq} : L \times L \to R^{\geq 0}$ represent the cost of assigning two different labels to adjacent nodes of edge $(p, q)$. Let $S : N \to L$ be the segmentation function that assigns to each node $p$ the segment label $S(p)$. The total cost of a segmentation S can be represented as

$$\sum_{p \in N} D_p(S(p)) + \sum_{p,q \in N} V_{pq}(S(p), S(q)). \tag{23.1}$$

While conceptually appealing, this objective function is fairly general and doesn't fully capture the specific requirements of the segmentation problem. Particularly, in the segmentation problem the rendering constraint applies, e.g., the area on the screen occupied by a child node is geometrically contained inside that of its parent node in the DOM tree. Therefore, any segmentation should follow the rule that if it places the root of a subtree in a particular segment, then it has to place all the nodes in the entire subtree in the same segment.

One relatively straightforward approach to incorporating the rendering constraint in the above objective function is based on the observation that the actual labels of segments do not really matter in segmentation – one mostly cares about whether adjacent nodes have the same label. Thus the labels can be treated as indistinguishable, and the objective function (23.1) can be changed to

$$\sum_{S(p) \neq S(q)} V_{pq} + \sum_{S(p) = S(q)} (1 - V_{pq}) \tag{23.2}$$

Using objective function (23.2), the rendering constraint can be enforced by considering only the leaf nodes of the DOM tree in constructing the weighted graph, and perform segmentation on these nodes only.

Equation (23.2) encodes the so-called correlation clustering problem, which has known practical combinatorial approximation algorithms, including a recent one called CClus by Ailon, Charikar, and Newman [2] that approximates Eq. (23.2) to within a factor 2.

Two main types of features similar to the ones used in the algorithms described in the previous sections are defined to compute the edge weights $V_{pq}$: visual and content based. The visual features include the node's position, shape (aspect ratio), background color, types, sizes, and colors of text. The content-based features are designed to capture the purpose of the content and include features such as average size of sentences, fraction of text within anchors, and tag names. The edge weights are derived from these features using a logistic regression function learned from a set of manually labeled Web pages, where each DOM node is assigned a segment ID. The logistic function takes the feature of two adjacent nodes as input and output the probability of the two nodes belong to the same segment, which is used as the $V_{pq}$ score.

While the correlation clustering formulation is intuitive and easy to implement, it has the drawback that it relies solely on the leaf nodes of the DOM tree. By design, the leaf nodes tend to carry limited information, which makes the features sparse and noisy, and thus negatively impact the segmentation quality. To address this problem, Chakrabarti et al. further proposed a different formulation based on energy-minimizing cuts, which is able to take into consideration all DOM nodes while preserving the rendering constraint [7]. While this approach is more general than the correlation clustering approach, it is also much more complex both in terms of graph construction and parameter training, and will not be covered in this chapter. Interested readers are referred to the original publication for more details on this algorithm.

## Web Content Extraction

The information amount in Web steadily grows to be the largest public knowledge repository. As the accumulation of Web pages, extracting Web data into database and spreadsheet instantly and accurately is crucial to diverse applications, such as product catalogue extraction or online price competition. Data in Web documents is usually extracted by wrappers, which are specialized program routines that can make the content of HTML pages directly available. Wrappers extract web contents in three steps: download HTML pages from a Web site, recognize and extract specified data, and save this data in a suitably structured format to enable further manipulation.

### Motivation and Challenges

Web content is the reason that people visit and browse the Web pages. Web pages contain not only the body contents, such as textual information or visual contents, but also many other distracting features including pop-up advertisements, scattered decoration images, or unnecessary hyperlinks. Automatic extraction of useful and relevant contents from Web pages has many applications, ranging from define "accordion summarization" to Web page classification and labeling, from enabling end users to view Web more freely over small screen devices such as personal digital assistants (PDAs) or cellular phones (e.g., ThunderHawk: http://www.bitstream.com/wireless/server/workflow.html), to provide better access to the Web for visually impaired and blinds. Typical tools and systems include Opera (http://www.openxml.org), Hal Screen Reader by Dolphin Computer Access (http://www.dolphinuk.co.uk/products/hal.htm), and Microsoft's Narrator (http://www.microsoft.com/technet/treeview/default.asp?url=/technet/prodtechnol/winxppro/reader_overview.asp).

Within the most straightforward Web document, HTML files, web content is displaying in diverse formats: textual, images, sounds, videos, or animations. Essentially, web contents can be classified into two types:

- Textual contents
  Textual contents include all the written contents on the page, both in text sections and inside figures. The best textual Web content is the text written for the Web directly, rather than simply copy and pasted from another print source.
- Multimedia contents
  The other type of Web content is multimedia content, which refers to any content that isn't the text. Multimedia contents include several different types of Web contents: animation, images, sound, video, etc.

  In order to analyze a Web page for content extraction, the typical method is writing specialized programs called wrappers/parsers to extract target information and contents and mapping them to new formats that support better data exchange and integration. Usually the Web content extraction contains the following steps:

- Retrieving and analyzing a Web page structure for content extraction
- Passing the page to a HTML parser
- Correcting HTML errors if necessary and converting web contents into XML or DOM tree representations for better understanding and organization
- Segmenting Web pages into different zones or categories, extracting the target contents accurately, and mapping them into a predefined data structure for further use.

### Web Content Extraction Methods

Since a web page usually contains not only the central content but also many auxiliary elements such as images and extraneous links, automatically identifying and extracting the target content to satisfy different research goals is vital to many applications. The most fundamental and challenging problem of Web content extraction is how to identify and extract the target contents needed.

HTML parser is widely used to extract contents from Web pages in either a linear or a nested fashion. Although different HTML parsers have different working schemes, they share the same input source – HTML pages – and parse Web contents into various formats, such as creating a Document Object Model (DOM) trees as [9] did or XML files, which reflect the HMTL tag hierarchy.

In the last decade, many works have been done on Web content extraction domain. At beginning, researchers started with the manual extraction and wrapper induction. Hammer et al. extracted semi-structured data from HTML pages and converted the information into database objects. Only heuristics based on text patterns is used to identify the boundary of the relevant data. McKeown et al. [36] used semantic boundaries to detect the largest body of the text on a Web page, but this straightforward method only works well for single-body documents with simple structures. Finn et al. [14] is another example that has a strict requirement on the Web page format. This method only extract Web contents from "single-article" sources, where content is presumed to be in a single body. Such manual-based methods can work well for large volume data with preknown document layouts. However, once the HTML page format is changed, the rules should be updated accordingly and are difficult to maintain and extend to new Web pages.

Later methods involving more advanced machine learning techniques were developed in the Web extraction process. A pioneer work for analyzing Web extraction tools and techniques is the RISE Web site (http://www.isi.edu/integration/RISE/) while a detailed survey and discussion can be found in [28]. Laender et al. classified the Web content extraction tools into the following categories:

- Languages for wrapper development: emphasizing the prevalent language to construct wrappers, such as Perl, PHP, or JAVA
- HTML-aware tools: taking advantage of Web page hierarchy by turning a Web page in a parsing tree. The representative tools include World Wide Web Wrapper Factory (W4F: http://db.cis.upenn.edu/DL/WWW8/index.html), XMRAP, and Lixto (www.lixto.com). Wrapper generation with Lixto is based on examples selected by users. All these tools share similar steps: identifying interesting regions on a HTML page, identifying important semantic tokens, and determining the nesting hierarchy of a page.
- NLP-based tools: building relationship between textual objects using filtering or part-of-speech tagging then deriving extraction rules; RAPIER [6] and WHISK [41] are representative NLP-based Web extraction tools, which apply multiple famous NLP techniques, such as filtering or part-of-speech tagging.
- Wrapper induction tools: generating rules based on training examples; WIEN [27] and SoftMealy are representative approaches that generate wrappers by learning from a set of real examples. Such methods can have good performance without many linguistic constraints. The quality of example pages is crucial to the performance of wrapper generation.
- Modeling-based tools: locating target data based on predefined models. Not surprisingly, researchers also tried to define models for Web pages based on the structure information and modeling primitives. NoDoSE [1] and DEByE [29] are two typical modeling-based Web content extraction tools.
- Ontology-based tools: relying directly on the data instead of structure features; Embley et al. [13] extracted Web objects based on ontology given in a specific domain. Such ontology-based methods can achieve a higher accuracy but the approach incurs high cost.
- Heuristic-based tools [17] is a newly published paper, which present an automatic approach to extract the main content of the Web page using tag tree and heuristics to filter the clutter and display the main content.

Although these methods vary at details, they fulfil a similar task: recognizing and extracting objects from a Web page and populating into a data repository. Based on various methods listed above, many researchers keep extracting Web contents in diverse ways: free-text searching, specific image collection (e.g., business logos or scientific charts), semi-structured document logical components (tabular structured data or scientific references), simplified document summarization, fact information detection supporting the quality of question-answering, document similarity analysis in both structure and semantic levels, etc. The methods to identify and extract a semi-structured document logical component – tables – from Web pages are elaborated in the next section.

### Web Table Extraction

Many powerful systems (e.g., OCTOPUS) use extracted Web tables as fundamental components for various further applications. Web table extraction usually contains two phases:

- Phase 1: Web table boundary detection and table rows/data cells identification
  - Table rows and data cells are delineated by the HTML markups such as <TABLE></TABLE> or <th>/</th>. Based on the tags, researchers usually convert the HTML table structure into a tree shape, which the children of each node are the next lower level of table elements.
  - Most HTML table detection relies on knowledge engineering techniques like heuristic rules based on the formatting cues and machine learning techniques (e.g., decision trees, Support Vector Machine, and wrapper learning, conditional random fields). In addition, they are highly domain specific.
    - Chen and Tseng [9] tested their algorithm based on heuristic rules and cell similarities on 918 HTML tables from airline information Web pages. Their table detection algorithm uses string, named entity and number category similarity to decide if it is a real or non-real table.
    - Hurst talked about a Naive Bayes classifier algorithm but there is no detail about the algorithm and the experiment [21]. They firstly mentioned the idea of rendering HTML code in order to use the results for detecting relational information in Web tables. Although they do not actually render documents in a browser, they can infer relative positional information of table nodes in an abstract table model.
  - Different from the traditional methods, visual rendition and topographical analysis is another representative method.
    - Kuprl et al. [26] used the visual rendition from the open-source Mozilla browser rather than the HTML tags to detect the Web tables. Their bottom-up clustering algorithm grouped visualized words into large groups and used a set of heuristics to detect tables unsupervised. The reported performance is about 70 %. In addition, there are several limitations: only those table types that are known in advance are supported by their algorithm. If there is a vertical gap in table cells, the algorithm currently does not deliver correct results. Also, the process is relatively slow, because it has to wait for Mozilla to render a page.
    - Gatterbauer and Bohunsky [16] provided an innovative approach to deal with the table extraction problem from Web pages by analyzing CSS2 visual box model. CSS represented textual HTML document elements by rectangular boxes, and a double topographical grid is adopted to manage all these visual boxes. A recursive spatial reasoning algorithm named VENTrec is proposed. VENTrec used keywords as input and tried to expand from any possible HyperBox on the grid into all directions until no possible expansion. The authors relied upon the positional and metadata information of visualized DOM element nodes in a browser. They believed

  that the individual steps of table understanding become easier by taking advantage of visual cues of the Web.

- Phase 2: Identifying true Web tables by filtering out false Web tables
  - Some researchers work on the classification of HTML tables because HTML tables are often used to format documents instead of presenting data. To decide whether a table is worthy of further process, we should combine language and layout information for a better content judgement. The drawback of the inconsistent HTML markups tags incurs the difficulty of the further identification on many table components, e.g., table header lines.
    - Wang and Hu concentrated on the HTML table detection in [42]. They classified HTML tables into two types: genuine tables and non-genuine tables, according to whether a two-dimensional grid is semantically significant in conveying the logical relations among the cells. They introduced 16 features that reflect the layout as well as content characteristics of tables. These features are trained on thousands of examples, which are used to build a large table ground truth database.
    - WebTable (http://windowsutilities.com/webtable.html) is a recent work with the largest scalability. It extracted 14.1 billion raw HTML tables from the English documents in Google's main index. In order to filter out poor-quality non-relational tables, a series of techniques are adopted to recover those real tables. Only 1.1 % of raw HTML tables are confirmed as relational tables (genuine tables). The resulting tables can be imported into a spreadsheet or database program.

Based on all the above table extraction works, we can observe and summarize the following trends: (1) the extensibility and scalability are two important evaluation parameters that obtain more and more attention. (2) As the prevalent of machine learning methods, they are also widely used to extract Web tables and the performance beats that of rule-based methods in most cases (▸Chap. 19 (Recognition of Tables and Forms)).

## Analysis of Text in Web Images

As the use of the Internet continues to grow and Web publishing technology advances, multimedia contents such as images contribute an increasingly more prominent proportion of the Web content. A survey by Petrie et al. [39] shows that among 100 home pages from 10 Web sites, there are on average 63 images per homepage. A significant portion of these images contains text and as such important semantic information about the documents in which the images are embedded.

### Motivation and Challenges

The need to embed text in images typically arises from the desire of the Web document author to present the text with an effect that is not possible using standard markup language features. This text is usually created with a graphics software tool, where the desired effects are applied to both the text and the background, and the result is saved in image format (e.g., JPEG, GIF). This means the embedded text is usually text of high importance such as titles and headings, because the authors have devoted significant effort to make them "stand out" from the surrounding text. Despite the existence of means for alternative textual descriptions of images (e.g., the ALT tag), Web authors typically do not ensure the existence or consistency of these descriptions. Thus, being able to identify and recognize text embedded in image form using document image analysis tools is crucial for unlocking the semantic content and making it available for important downstream analyses.

The recognition of text in image form on the Web presents unique challenges. On the one hand, such text is free from variations and noises introduced in the scanning process. On the other hand, it has other properties that make it more difficult to process than scanned text encountered in traditional OCR. First, text images from the Web tend to be color images of very low resolution (e.g. 72 dpi) and the font size used for text is often very small (e.g. 5–7 pt). Such conditions pose a challenge to traditional OCR, which typically works with 300 dpi images that are mostly bi-level and character sizes of usually 10 pt or larger. Furthermore, images on Web documents tend to have various artifacts introduced by the authoring software. Finally, these images are often highly complex, with color variation not just in the background but in the foreground characters as well.

While some of these issues are shared by the related problem of recovering text from video sequences or natural scenes (▶Chap. 25 (Text Localization and Recognition in Images and Video)), text recovery from Web images faces unique challenges. The recovery of text form video can leverage the temporal continuity of video frames. For example, text in adjacent frames tends to be in identical or similar color. In text extraction from natural scenes, a common assumption that the text as well as the immediate background of uniform color. This assumption does not apply for most Web text images.

## Approaches

The task of recognizing text in images can be decomposed into two main steps: (1) text isolation (also called localization), where the image is segmentated such that regions corresponding to potential text components are separated from the background, and (2) text recognition, where strings of character-like components are recognized.

It should be clear that the unique challenges facing text recognition from Web images as identified above mostly affect the first stage of this process. Once this first stage is carried out, techniques from traditional OCR can be adapted with relative ease to carry out the rest of the task. Thus, the focus here will be on methodologies for text isolation from Web images.

### Methods Based on Traditional Color Space

Early approaches to text isolation relied largely on traditional clustering and quantization techniques. Lopresti and Zhou [32] proposed methods for text segmentation, extraction, and recognition. Their method for text segmentation and extraction is based on clustering in the RGB color space and then for each cluster assessing connected components whose color belongs to that cluster. The approach works well with GIF images (only 256 colors) and when characters are of almost uniform color. With similar assumptions, the segmentation approach of Antonacopoulos and Hu [4] uses two alternative clustering approaches in the RGB space but works on (bit-reduced) full-color images (JPEG) as well as GIFs. Jain and Yu [22] proposed segmentation method based on decomposing an original image into a number of foreground images and a background one. The original number of colors (8- or 24-bit images) is reduced a to a much smaller number (between 4 and 8) of distinct colors by bit dropping and color quantization in the RGB space. Although this method produces good results for relatively simple images, it shares the same problems with the other methods for more complex images.

### Methods Based on the Perceptual Color Space

A drastically different approach based on perceptual color decomposition was proposed by Karatzas and Anotnacopoulos in [24]. This approach adopts a segmentation method based on analyzing differences in chromaticity and lightness that are closer to how humans perceive distinct objects, and has been demonstrated to work effectively under conditions of varying foreground and complex background.

The approach comprises four main steps. Note that the HLS (hue, Lightness, and saturation) color system is used throughout this approach. HLS is a perceptually oriented system that is more suitable for the analysis for images created to be viewed by humans than the traditional RGB system. The four steps are summarized below. Interested readers are referred to the original publication [24] for more details.

- Step 1. Chromatic/achromatic layer splitting
  The first step is a preprocessing step based on information provided in published studies on color discrimination by humans in terms of expressions of minimum discrimination ability as a function of each of the physical properties of color (wavelength, color purity, and luminance) [45]. This step separates the chromatic from the achromatic content of the image for independent further processing. Chromatic color is any color for which a dominant wavelength can be identified (e.g., red, green, blue, yellow). A color is said to be achromatic if no dominant wavelength can be identified, i.e., shades of grey including black and white. Separating chromatic from achromatic pixels at this stage is important because any process that examines hue values cannot be successfully applied to achromatic pixels.
- Step 2. Further splitting
  In this step subsequent splitting process is applied to identify and describe areas of perceptually similar color in the image. The chromatic and achromatic layers are both split into a number of layers with more refined color uniformity using global information derived from the hue or lightness histogram of each layer.
  For the pixels of the achromatic layer, the histogram of lightness is computed and peaks are identified. If two or more peaks are present, consecutive peaks are analyzed by examining the horizontal distance (lightness value difference) between them and their corresponding height difference (ratio of peak maxima),

using thresholds determined in [45] and [23]. At the end of this peak analysis and combination process, the pixels in the layer that have lightness values under each final peak group are exported to a separate sub-layer.

For chromatic colors, first the histogram of the hue values is computed for the pixels of the chromatic layer and peaks are identified (in the same way as peaks of the lightness histogram in the achromatic layer). The horizontal distance (hue difference) between consecutive peaks and their corresponding difference in height is calculated and analyzed for potential combining using "similarity" criteria defined in [45] and [23]. The chromatic layer is thus split into sub-layers of different hue (each layer containing the range of hues under each of the final peaks). For each of the sub-layers produced, the lightness histogram is then computed, peaks are identified, and the peak analysis and grouping process is performed (as in the achromatic layer). New image sub-layers are created for pixels with lightness values in the ranges under each of the final peak groups.

A tree representation is produced after the splitting of the image into achromatic and chromatic layers and their corresponding sub-layers. In this tree, the root represents the original image and the nodes correspond to each of the layers produced by the splitting process. Each layer contains regions with a perceptually distinct hue or lightness.

- Step 3. Color-connected component analysis

  To prepare for merging (Step 4), connected components are identified on each leaf layer, using a one-pass labeling algorithm [4] and the average color of each component is computed and stored. Each connected component corresponds to a region in the image that has an effectively distinct color. To identify these components, first the component *extensibility* is examined based on color similarity. All pixels that satisfy the relevant color similarity criterion are noted as the *vexed* area of that component. The topological relationship between any two components is then examined by checking the overlap between the two components when the corresponding vexed areas are taken into consideration. This lead to a measurement of overlap called *overlapping degree*.

- Step 4. Merging

  In this final step, every possible pair of components in the leaf layer is examined, and if their overlapping degree is above a predefined threshold (experimentally derived 0.56), a possible merger is identified. All identified possible mergers in each layer are kept in a sorted list and merging starts with the pair of components with the highest overlapping degree. When two components are merged, the two core regions are combined to form the core region of the new component.

  After each merger is completed, other potential mergers involving one of the two no-longer-existing components with an existing component are reassessed to take into account the resulting newly created component. After all possible mergers have taken place within each of the leaf layers, merging between the components of all sibling layers (across the same level) is performed. The merging process is repeated level by level, moving up in the tree structure until the root of the tree (the original image) is reached.

  At the end of this four-step approach, the original image has been segmented into a number of connected components based on perceptual similarity.

These connected components can then be input to various traditional OCR engines for text line extraction and ultimately text recognition.

While this perceptual-based approach shows great promise, the evaluation of this method remains limited. Downstream analysis methods are needed to fully connect this approach, the state of the art in text line extraction and recognition, so that a complete system of text extraction and recognition applied to general images can be tested and eventually deployed for real applications.

## Web Document Modeling

An area closely related to Web content extraction is Web document modeling, e.g., the modeling of the geometric relationships among different components of various Web documents. While a significant amount of work has been dedicated to representing the geometric structure of a scanned document (▸Chap. 6 (Analysis of the Logical Layout of Documents)), relatively little has been done to address the specific characteristics of Web documents. These special characteristics include:

1. The layout of a Web document is specified in the Cascading Style Sheet (CCS). However the CCS model is designed to be used by the Web browser's layout engine for the sole purpose of generating the visual representation of a Web page. It cannot be used to describe spatial relationships and configurations of the various visual elements.
2. A Web document often includes a much larger variety of objects than a scanned document. These objects may include navigation menu, Web form elements, and posts in a Web forum. As such, a Web document could be not only a resource for textual or media information but also an application.

Since work in this area is still in its very early stages and the definitive technique has yet to emerge, a survey of the exiting work and pointers to a promising new direction are provided here.

Most of the work on representing geometric structure of a Web document has been carried out in a largely ad hoc manner, driven by the specific applications including predominantly Web content extraction and Web content classification. The most widespread representation is based on X-Y cut type of document segmentation as described in Section "Web Page Segmentation", an approach borrowed from scanned documents. This leads to a layout model based on inclusion relations, where the whole document can be represented as a tree, and has been used primarily for content extraction and retrieval applications [12, 16]. Another type of representation is based on adjacent graphs such as 2D conditional random fields used for content extraction [48]. Finally, in some methods the DOM tree is augmented with visual characteristics to indirectly represent the geometric relationships for content extraction [34].

The first attempt at constructing a comprehensive geometric model of Web documents has just been reported recently. The authors propose a very general modeling framework called Web Page Geometric Model (WPGM). The model takes CSS boxes and their features as computed by the browser's layout engine as the main building blocks, called Geometric Objects (GO), and provides mechanisms for both *quantitative modeling* (i.e., spatial position attribute) and *qualitative modeling* (i.e., spatial relationship among objects). The spatial relations supported include topological relations [11], distance between objects, directions, alignment [25], and 2D interval relations. Originally designed for Web accessibility, this approach has the potential of providing for the first time a unifying model that can be used for various applications including content extraction, Web page classification, and customized rendering. However its effectiveness and utility remains to be tested in these broad applications.

## Available Software

In this section, we introduce multiple representative open-source or business software and tools, which can analyze digital documents in PDF or HTML format. All these tools provide high-quality data sources and information foundation for further document analysis and understanding.

### PDF Analysis Tools

Since most PDF documents describes the low-level structural objects such as a group of characters, lines, curves, and images as well as associated style attributes such as font, color, stroke, fill, and shapes, accurately extracting all the objects with minimal potential component overlay and the interference between different logical components is a challenging problem. Both open-source and commercial converters are available. These converters usually only use PDF's low-level-based features due to PDF's complex generation process. The major state-of-the-art tools for PDF analysis and starting with those professional packages and libraries are analyzed in this section. All the listed tools are mature tools that are freely accessible and can be used directly.

- Adobe Systems Inc. provides a software developers kit (SDK) to extract logical words from a PDF file. This SDK gives access via the API of Acrobat viewers to the underlying portable document model. Within the viewers, there is a built-in algorithm that finds all the logical words within a PDF document for string-search purposes. This algorithm is sound for all single-column documents but can get confused when presented with multicolumn texts.
- JPedal (http://www.jpedal.org/), a library for Java development of PDF applications, has both commercial and open-source versions. End users submit PDF documents in the JPedal Web site and JPedal extracts text, images, annotations, and forms and notifies users the result of the extraction in XML format via emails. JPedal made good attempt on reading order resorting task.
- MatterCast (http://www.mattercast.com/default.aspx) has a commercial product called SVG Imprint. MatterCast can convert PDF to SVG in a single or batch way. The output in SVG can retain the original document layout and integrates all the

fonts embedded in PDF document. The raster images inside the PDF documents are extracted as separate file.

- BCL (http://www.bcltechnologies.com/document/index.asp) provides three products BCL Drake, BCL Freebird, and BCL Jade. All of them have either in input or output PDF documents. BCL Drake converts PDF documents into RTF for viewing and editing in Microsoft Word. BCL Freebird, as a plug-in for Adobe Acrobat, can convert PDF documents into graphics (Tiff, Jpeg, and Bmp). As another Acrobat plug-in, BCL Jade can extract text, tabular data, and graphics from PDF documents. However, this work can only be finished semi-automatically, with the help of mouse selection. Instead of being able to extract the images embedded inside a PDF page, BCL can only extract the image of the whole page.
- Glance (http://www.pdf-tools.com/en/home.asp) is another commercial PDF analysis product with a set of products and API. All of them have either in input or output PDF documents. The products include pdimgcomp (a PDF image compression, conversion, and extraction tool), Img2pdf and text2pdf (two products that convert images and text into PDF documents, respectively), pdcat useful for concatenation, pdsplit for splitting, pdsel for page selection, pdw for text extraction in ASCII or Unicode, and pdwebl for adding link annotations.
- Apache PDFBox$^{TM}$ (http://pdfbox.apache.org/) is an open-source Java PDF library for working with PDF documents. This project allows creation of new PDF documents, manipulation of existing documents, and the ability to extract content from documents.

## Web Document Analysis Tools

HTML parsers are implemented in different languages such as Java, Perl, or Python. The typical tools in Java include HTMP parser in http://htmlparser.sourceforge.net/ and TagSoup in http://java-source.net/open-source/html-parsers/tagsoup.

The former introduces a fast JAVA library of real-time parser for real-world HTML pages, which supports two main functionalities: Web content extraction and transformation. Extraction encompasses all the information retrieval programs such as text extraction, link extraction, screen scraping, image/sound extraction, and other affiliated functions (link check as well as site monitoring). HTML parser produces a stream of tag objects, which can be further parsed into a searchable tree structure.

HtmlCleaner (http://htmlcleaner.sourceforge.net/) is another open-source HTML parser written in Java. It is designed to clean up the mess introduced by the original HTML page authors and reorganize the texts, attributes, and tags nicely. Similar to other HTML parsers, HtmlCleaner accepts HTML documents as input files. After processing and dirty cleaning, a well-formed XML file is generated.

NekoHTML (http://nekohtml.sourceforge.net/) is a similar HTML parser with cleaning and rendering functions. It scans and parses HTML files, fixes many common mistakes or unsuitable contents, then displays useful information using standard XML interfaces. In addition, NekoHTML can fix the tag unbalance problem by providing custom tag and adding missing parent elements.

Other similar tools include HotSAX (http://hotsax.sourceforge.net/), Cobra 0.95.1 (http://html.xamjwg.org/cobra.jsp), JTidy (http://jtidy.sourceforge.net/), and VietSpider HTMLParser (https://sourceforge.net/projects/binhgiang/).

## Conclusion

This chapter provides a detailed account of the challenges, approaches, and the state of the art for the analysis of documents that originate from a digital form. The focus is on PDF and Web documents, two of the most dominate forms of digital document encoding. Challenges for both types of documents are centered around extraction of basic content elements, understanding of the layout and relationship among these elements, and organization of these elements into higher-level, semantically meaningful segments suitable for downstream applications such as information extraction and archiving, Web mining, and Web content repurposing. Addressing these challenges require a combination of advanced techniques drawn from the diverse fields of machine learning, computational linguistics, information retrieval, and image processing. Significant progress had been made in the last 15 years and many tools are now available that provide basic functionalities. However, challenges remain, particularly around the areas of extraction of complex basic elements (e.g., tables, text embedded in images), the understanding of pages with complex layout structures, and system extensibility and scalability for the efficient and effective handling of large number of documents. Some new directions have emerged, including perceptually based methods, graph theoretic approaches, and generative models. The full development and evaluation of these approaches provide significant and promising scope for future research.

## Cross-References

▸Analysis of the Logical Layout of Documents
▸Page Segmentation Techniques in Document Analysis
▸Recognition of Tables and Forms
▸Text Localization and Recognition in Images and Video

## References

1. Adelberg B (1998) NoDoSE – a tool for semi-automatically extracting structured and Semi-structured data from text documents. In: ACM SIGMOD international conference on management of data (SIGMOD'98), Seattle, pp 283–294
2. Ailon N, Charikar M, Newman A (2005) Aggregating inconsistent information: ranking and clustering. In: 37th STOC, Baltimore, pp 684–693

3. Anjewierden A (2001) AIDAS: incremental logical structure discovery in PDF documents. In: 6th international conference on document analysis and recognition (ICDAR), Seattle, Sept 2001, pp 374–378
4. Antonacopoulos A, Hu J (ed) (2004) Web document analysis: challenges and opportunities. World Scientific, Singapore
5. Cai D, Yu S, Wen J-R, Ma W-Y (2003) Extracting content structure for web pages based on visual representation. In 5th Asia Pacific Web Conference, pp 406–415
6. Califf ME, Mooney RJ (1999) Relational learning of pattern-match rules for information extraction. In: Proceedings of the sixteenth national conference on artificial intelligence and the eleventh innovative applications of artificial intelligence conference, AAAI'99/IAAI'99, Orlando. Menlo Park, pp 6–11
7. Chakrabarti D, Kumar R, Punera K (2008) A graph-theoretic approach to webpage segmentation. In: WWW 2008, Beijing, pp 377–386
8. Chao H, Fan J (2004) Layout and content extraction for pdf documents. In: Marinai S, Dengel A (eds) Document analysis systems VI. Lecture notes in computer science, vol 3163. Springer, New York/Berlin, pp 13–224
9. Chen JS, Tseng DC (1996) Overlapped-character separation and construction for table-form documents. In: IEEE international conference on image processing (ICIP), Lausanne, pp 233–236
10. Chen Y, Xie X, Ma W-Y, Zhang H-J (2005) Adapting web pages for small-screen devices. Internet Computing, 9(1):50–56
11. Cohn AG (1997) Qualitative spatial representation and reasoning techniques, vol 1303. Springer, Berlin, pp 1–30
12. Duygulu P, Atalay V (2002) A hierarchical representation of form documents for identification and retrieval. IJDAR 5(1):17–27
13. Embley DW, Campbell DM, Jiang YS, Liddle SW, Lonsdale DW, Ng Y-K, Smith RD (1999) Conceptual-model-based data extraction from multiple-record web pages. Data Knowl Eng 31(3):227–251
14. Finn A, Kushmerick N, Smyth B (2001) Fact or fiction: content classification for digital libraries. In: Joint DELOS-NSF workshop on personalisation and recommender systems in digital libraries, Dublin, p 1
15. Futrelle RP, Shao M, Cieslik C, Grimes AE (2003) Extraction, layout analysis and classification of diagrams in PDF documents. In: International conference on document analysis and recognition (ICDAR) 2003, proceedings, Edinburgh, vol 2, p 1007
16. Gatterbauer W, Bohunsky P (2006) Table extraction using spatial reasoning on the CSS2 visual box model. In: Proceedings of the 21st national conference on artificial intelligence (AAAI), Boston, vol 2, pp 1313–1318
17. Gupta N, Hilal S Dr (2011) A heuristic approach for web content extraction. Int J Comput Appl 15(5):20–24
18. Hadjar K, Rigamonti M, Lalanne D, Ingold R (2004) Xed: a new tool for extracting hidden structures from electronic documents. In: Document image analysis for libraries, Palo Alto, pp 212–224
19. Hassan T (2009) Object-level document analysis of PDF files. In: Proceedings of the 9th ACM symposium on document engineering (DocEng'09), Munich. ACM, New York, pp 47–55
20. Hassan T (2009) User-guided wrapping of PDF documents using graph matching techniques. In: International conference on document analysis and recognition – ICDAR, Barcelona, pp 631–635
21. Hurst M (2001) Layout and language: challenges for table understanding on the web. In: Proceedings of the 1st international workshop on web document analysis, Seattle
22. Jain AK, Yu B (1998) Automatic text location in images and video frames. Pattern Recognit 31(12):2055–2076
23. Karatzas D (2002) Text segmentation in web images using colour perception and topological features. PhD Thesis, University of Liverpool

24. Karatzas D, Anotnacopoulos A (2007) Colour text segmentation in web images based on human perception. Image Vis Comput 25(5):564–577
25. Kong J, Zhang K, Zeng X (2006) Spatial graph grammars for graphical user interfaces. CHI 13:268–307
26. Krupl B, Herzog M, Gatterbauer W (2005) Using visual cues for extraction of tabular data from arbitrary HTML documents. In: Proceedings of the 14th international conference on world wide web (WWW), Chiba
27. Kushmerick N (2000) Wrapper induction: efficiency and expressiveness. Artif Intell Spec Issue Intell Internet Syst 118(1–2):15–68
28. Laender AHF, Ribeiro-Neto BA, da Silva AS, Teixeira JS (2002) A brief survey of web data extraction tools. ACM SIGMOD Rec Homepage Arch 31(2):84–93
29. Laender AHF, Ribeiro-Neto B, da Silva AS (2002) DEByE – date extraction by example. Data Knowl Eng 40(2):121–154
30. Lien Y-LL (1989) Apparatus and method for vectorization of incoming scanned image data. United States Patent US4,817,187, assigned to GTX Corporation, Phoenix, Arizona, 28 Mar 1989
31. Liu Y, Bai K, Mitra P, Lee Giles C (2007) TableSeer: automatic table metadata extraction and searching in digital libraries. In: ACM/IEEE joint conference on digital libraries, Vancouver, pp 91–100
32. Lopresti D, Zhou J (2000) Locating and recognizing text in WWW images. Inf Retr 2(2/3):177–206
33. Lovegrove W, Brailsford D (1995) Document analysis of PDF files: methods, results and implications. Electron Publ Orig Dissem Des 8(3):207–220
34. Luo P, Fan J, Liu S, Lin F, Xiong Y, Liu J (2009) Web article extraction for web printing: a DOM+visual based approach. In: Proceedings of the DocEng, Munich. ACM, pp 66–69
35. Marinai S (2009) Metadata extraction from PDF papers for digital library ingest. In: Proceedings of the 10th international conference on document analysis and recognition (ICDAR), Barcelona, pp 251–255
36. McKeown KR, Barzilay R, Evans D, Hatzivassiloglou V, Kan MY, Schiffman B, Teufel S (2001) Columbia multi-document summarization: approach and evaluation. In: Document understanding conference, New Orleans
37. Okun O, Doermann D, Pietikainen M (1999) Page segmentation and zone classification: the state of the art. Technical report: LAMP-TR-036/CAR-TR-927/CS-TR-4079, University of Maryland, College Park, Nov 1999
38. Oro E, Ruffolo M (2009) PDF-TREX: an approach for recognizing and extracting tables from PDF documents. In: ICDAR'09 proceedings of the 2009 10th international conference on document analysis and recognition, Barcelona, pp 906–910
39. Petrie H, Harrison C, Dev S (2005) Describing images on the web: a survey of current practice and prospects for the future. In: Proceedings of human computer interaction international (HCII), Las Vegas, July 2005
40. Smith PN, Brailsford DF (1995) Towards structured, block-based PDF. Electron Publ Orig Dissem Des 8(2–3):153–165
41. Soderland S, Cardie C, Mooney R (1999) Learning information extraction rules for semi-structured and free text. Mach Learn Spec Issue Nat Lang Learn 34(1–3):233–272
42. Wang Y, Hu J (2002) Detecting tables in HTML documents. In: Fifth IAPR international workshop on document analysis systems, Princeton, Aug 2002. Lecture notes in computer science, vol 2423, pp 249–260
43. Wang Y, Phillips IT, Haralick RM (2000) Statistical-based approach to word segmentation, In: 15th international conference on pattern recognition, ICPR2000, vol 4. Barcelona, Spain, pp 555–558
44. Wasserman HC, Yukawa K, Sy BK, Kwok K-L, Phillips IT (2002) A theoretical foundation and a method for document table structure extraction and decomposition. In: Lopresti DP, Hu J, Kashi R (eds) Document analysis systems. Lecture notes in computer science, vol 2423. Springer, Berlin/New York, pp 29–294

45. Wyszecki G, Stiles W (1982) Color science: concepts and methods, quantitative data and formulae, 2nd edn. Wiley, New York
46. Yildiz B, Kaiser K, Miksch S (2005) pdf2table: a method to extract table information from PDF files. In: Proceedings of the 2nd Indian international conference on artificial intelligence (IICAI05), Pune, pp 1773–1785
47. Zanibbi R, Blostein D, Cordy JR (2004) A survey of table recognition: models, observations, transformations, and inferences. Int J Doc Anal Recognit 7(1):1–16
48. Zhu J, Nie Z, Wen J-R, Zhang B, Ma W-Y (2005) 2D conditional random fields for web information extraction. In: Proceedings of the ICML'05, Bonn. ACM, pp 1044–1051

## Further Reading

Web Document Analytics, Apostolos Antonacopoulos and Jianying Hu (Editors), World Scientific, 2004.
PDF Explained, John Whitington, O'Reilly Media, 2011.
Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data, Bing Liu, Springer, 2011.