Intelligent Wrapping from PDF Documents[⋆]

Tamir Hassan and Robert Baumgartner

Database and Artificial Intelligence Group, Institute of Information Systems,
Technische Universität Wien, Favoritenstraße 9-11, 1040 Wien, Austria
{hassan, baumgart}@dbai.tuwien.ac.at

Abstract. Wrapping is the process of navigating a data source, semi-automatically extracting data and transforming it into a form suitable for data processing applications. The semi-structured form of web pages, coupled with the availability of business-relevant data, has led to the availability of several established products on the market for wrapping data from the Web. One such approach is the *Lixto* methodology [1], a result of research performed at DBAI.
Many commercial applications also require the extraction of data from PDF documents. There appear to be no general-purpose approaches to fulfil this need and, as the PDF format is unstructured, this is a challenging task. We are investigating PDF data extraction in the NEXTWRAP project. This paper presents our work in progress, with particular reference to low-level segmentation algorithms.

1 Introduction

A wide variety of information on today's Web is published in Adobe's Portable Document Format (PDF). In particular, many business documents, such as financial reports, newsletters and patent applications, are commonly published in PDF.

The success of PDF can be attributed to its roots as a page-description language. Any document can be converted to PDF as easily as sending it to the printer, with the confidence that the formatting and layout will be preserved when it is viewed or printed across different computing platforms.

Unfortunately, this approach presents one major drawback: most PDFs contain little or no explicit structural information, making automated machine processing and data extraction a difficult task. Although later versions of the PDF specification support the use of XML tags to denote logical elements, these are seldom found in business documents.

When a human reader views the document, various *layout conventions* indicate to him/her its logical structure. In order to make PDF files amenable

[⋆] This work is partly funded by the Austrian Federal Ministry for Transport, Innovation and Technology under the FIT-IT programme line as a part of the NEXTWRAP project.

to machine processing, this logical structure needs to be rediscovered from the layout, typographical and textual features of the content itself. This process is known as *document understanding.*

2 Previous Work

2.1 Web Data Extraction

Current approaches to wrapper generation focus on semi-structured data sources such as HTML. One approach, the *Lixto Visual Wrapper*, allows a non-expert user to create wrapper programs in a predominantly visual and interactive fashion by clicking on example instances on a visual rendition of the web page. In the background, the software locates the data using a hierarchical representation of the web page, the HTML parse tree. The user can fine-tune the selected data by adding or removing logical conditions. The system then generates a program in *Elog* [1], a declarative logic-based language, to automatically extract this data from similarly structured sources, or from sources whose content changes over time.

2.2 Document Understanding

Much work in the document analysis community is concerned with processing documents that have been scanned or otherwise digitized. There are various methods of segmentation [5, 6] and classification [6] that work on a binarized image as input. Whilst we could make use of these algorithms by rasterizing the PDF, this would effectively be taking a step backwards. The object information that is available from the PDF file is already at a higher level; for example, text is already classified as such. Therefore, our approach is to segment the page directly on the object data (see section 4).

There is also significant work, particularly in the area of understanding tabular data, which deals with documents in monospaced ASCII format [4, 7]. These techniques do not always adapt well to PDF files, as PDFs make use of a much wider range of layout conventions to denote the same structural elements. We have implemented a variant of the whitespace density graph in [4] and this has proved useful at several levels of the document understanding process (see section 4.3).

In contrast, there has been considerably less work that has dealt directly with PDF files as input. One notable example is [2], in which the authors accessed the individual objects using the Adobe Acrobat API. Acrobat's inbuilt line-finding algorithm was used to access the line objects directly. With a number of counting and sorting procedures on the co-ordinates of each line, this method was able to discern considerable logical information about the PDF file.

The fact that the methods in [2] analysed only text, and ignored elements such as lines, boxes and images, suggests that the authoring process of a document often encodes logical structure in a redundant way, and that certain elements serve only to reinforce the structure and make it clearer to the reader.

Nevertheless, we plan to make use of all the information available to us in order to reliably understand as broad a class of documents as possible.

3 Overview of the Project

The first step in the extraction process is to segment, or break down, the document into blocks that can be said to be *atomic*, i.e. to constitute the smallest logical entity in the document. Typically these consist of paragraphs, headings, titles and captions. In order to generate as complete an understanding as possible, we also aim to analyse graphical elements on the page such as images, lines and boxes. We have experimented with some algorithms for this process, and they are described, together with preliminary results, in section 4.

This stage of the work is being undertaken in collaboration with the *AllRight* project at our institute, which aims to make use of similar visual techniques in extracting tabular data from web pages [3].

Once the low-level analysis is complete, we can begin to group these atomic blocks into larger *composite* structures and generate the logical structure of the document. This will enable us to extract data from the document in a logical way. We plan to investigate various methods for doing this, as summarized below. The process is illustrated in figure 1.

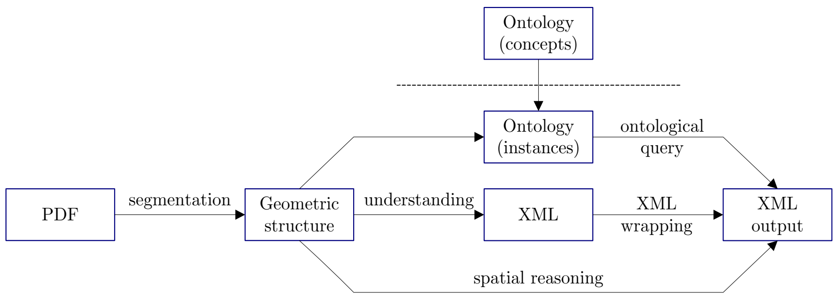


Fig. 1. Outline of possible methods for wrapping from PDF

- **ontology-based wrapping:** this approach makes use of a document-generic ontology to represent the rules and relationships between the various document objects. At some stage of the document understanding process, this ontology is populated with objects from the document. The remaining high-level understanding and content extraction is performed using established tools for ontological reasoning. This approach also can naturally be extended to ontologies that are specific to a particular document class.
- **conversion to a structured format:** this approach makes use of rules, expressed in a logical or procedural language, to classify and find relationships between the blocks. This process will allow us to represent the content

in a hierarchical structure such as XML. Thus the current techniques within the Lixto suite for wrapping from HTML could be extended to work with this XML format.
- **spatial reasoning:** this approach does not aim to generate a complete understanding of the document, but rather to select document objects according to how they occur (or relate to other objects) on the physical page. Sometimes, it may be simpler for a wrapper designer to sidestep the document understanding process and extract objects simply based on their location.

The final stage is to integrate the new methods with the existing Lixto software, ensuring that the user can interact with a visual rendition of the PDF file, with the complexity of the underlying representation being entirely hidden.

4 The Segmentation Process

The aim of segmentation is to divide the page into atomic segments, or blocks, that can be said to contain one logical entity, or "idea", in the document's structure. This is a task that utilizes visual cues on the page, and therefore borrows many techniques from computer vision. Naturally, there are two main approaches to segmenting a page; top-down and bottom-up. We have implemented both approaches, and our algorithms are described in sections 4.1 and 4.2. Section 4.3 describes some of the techniques we have used in our algorithms.

These algorithms were implemented in our prototype using PDFBox,[1] a Java LGPL library, to parse the PDF data and return a set of low-level page objects such as text fragments, lines, rectangles and images. The results of our processing are output to XML and are visualized using the XMIllum framework.[2]

4.1 Top-Down Hierarchical Segmentation

In the top-down approach each page is divided, usually hierarchically, into increasingly smaller segments until it is either not possible to divide the segment any further or it is determined that the segment contains one logical entity.

Our algorithm is based on [5]. The page is recursively divided by examining the whitespace density graph (see section 4.3) in both horizontal and vertical directions and looking for a region with 100% whitespace. If there is more than one region with 100% whitespace, the division with the highest *status*, or importance, must be chosen. The page is consequently divided into two regions by the chosen whitespace division and the process is repeated recursively for each region until no potential division with 100% whitespace is found. An example of this method is given in figure 2.

The benefit of this approach is that the resulting hierarchical structure can easily be adapted to represent the logical structure of the page by combining levels where the division or "cut" is made in the same direction. However, this approach fails to completely segment certain layouts, as shown in figure 3.

[1] PDFBox, `http://www.pdfbox.org`
[2] XMIllum, `http://xmillum.sourceforge.net`

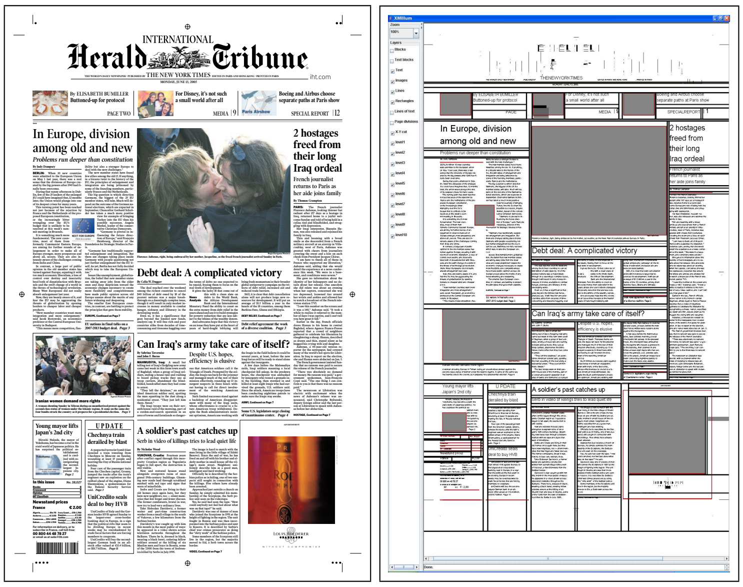


Fig. 2. Front page from the *International Herald Tribune* newspaper (left) with its successful top-down segmentation (right).

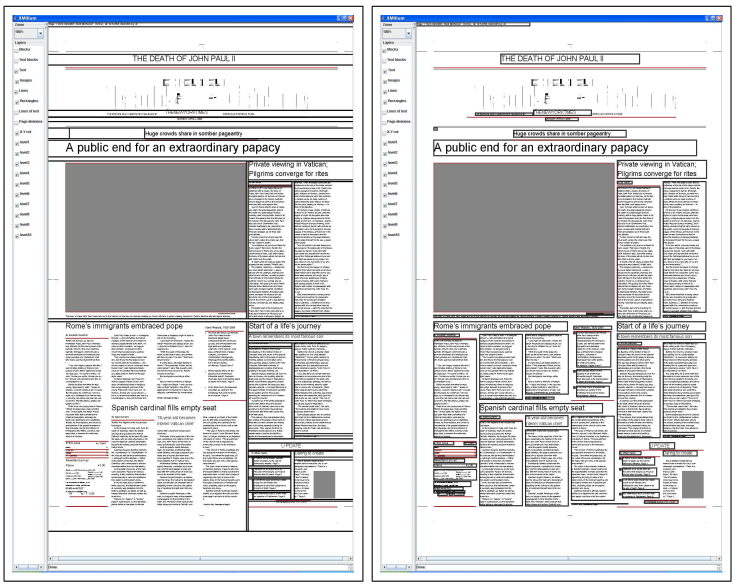


Fig. 3. A different issue of the *International Herald Tribune*. Top-down segmentation (left) fails to completely segment the page (the bottom-left quadrant is not segmented), whereas bottom-up clustering (right) succeeds.

4.2 Bottom-Up Clustering

In the bottom-up approach individual text fragments are clustered together to form increasingly larger and more meaningful blocks in the document. For example, text fragments are first merged into lines which, in turn, are merged into paragraphs.

Our algorithm uses heuristics to merge individual text fragments into lines. Each line is then examined in turn, from top to bottom. In a multi-column page each column is analysed separately. Lines are merged into complete paragraphs if the distance between them is below a certain threshold and if the font sizes are similar. This threshold distance is a function of the modal line spacing. The result is shown in figure 3, compared with the unsuccessful result of the top-down approach.

4.3 Techniques We Have Applied

The Neighbourhood Graph. In order to reduce the run-time of our bottom-up clustering algorithm we decided to store the content in an undirected graph where each text fragment (and higher-level composite objects) is represented by a node. All neighbouring objects within "line-of-sight" are connected by vertices. This data structure also has the benefit of simplifying the code in our clustering algorithm. An example is given in figures 4–6.

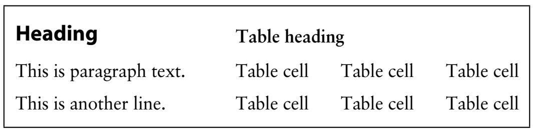
Page Divisions. A complex page layout, such as that of a newspaper, makes use of three predominant features to inform the reader where the various sections lie: rivers of whitespace, ruling lines and rectangular boxes. In our implementation, these separators are all represented as *page divisions*. In fact, lines and boxes are often redundant as their removal will leave a river of whitespace in their place. Therefore, it should be possible to locate such page divisions simply by analysing the text objects on the page. However, we plan to make use of the existence and thickness of ruling lines in determining the relative *status*, or importance, of each page division.

The Whitespace Density Graph. The *whitespace density graph* is a projection profiling method that scans along a given region of the page in a horizontal or vertical direction. Each point in the graph represents the total density of whitespace at that particular horizontal or vertical projection.

This term was introduced in [4], where a horizontal projection profile was used to determine the location of individual columns in tables. In [5] this method was used in both vertical and horizontal directions in the implementation of the *recursive* X-Y *cut* algorithm for segmenting a page. A variant of this algorithm has been implemented here.

Traditionally, projection profile methods worked on the individual pixel level to calculate the density at a given position. With PDF, it is not convenient to do this, and we approximate by assuming all objects to have uniform black density.

This technique also brings significant speed improvements [5] whilst producing a sufficiently accurate result for our purposes.



Heading **Table heading**

This is paragraph text. Table cell Table cell Table cell

This is another line. Table cell Table cell Table cell

Fig. 4. A simple example of paragraph text and tabular content on a page

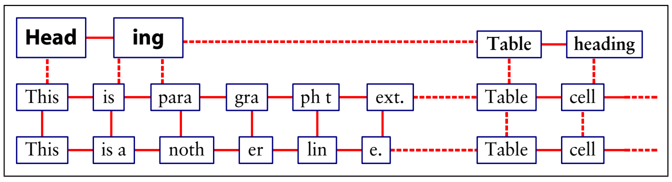


Fig. 5. Neighbourhood graph of figure 4. Vertices that join text fragments within the same segment are drawn with solid lines

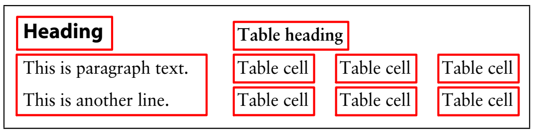


Fig. 6. The content of figure 4 after segmentation

5 Conclusion

Low-level page segmentation forms the basis upon which higher-level methods of document understanding and concept-based data extraction are performed. In this paper we have described two approaches for performing this process and their relative merits and disadvantages.

What makes the human visual system particularly good at understanding a page layout is its ability to analyse the image at different levels of granularity at the same time. For example, if we were to look at a newspaper page from a distance, we may not be able to see the individual words, but we can still tell where the columns lie. If we move closer to the page, we begin to notice the individual paragraphs, bylines, captions and other elements. Yet we can still tell where the individual columns lie, as the overall shape of the text dictates this to us.

We have tried to simulate some of these processes with our algorithms, and believe that the results can be improved further by combining our top-down and bottom-up approaches to layout analysis.

6 Further Work

We are currently experimenting with a more complex algorithm that aims to simulate more closely how a human reader would analyse a page. It is based on two principles:

- **Reasoning with uncertainty:** the use of a probabilistic or other score-based system to generate *confidence measures* to represent how likely a given decision is true or false
- **Reasoning at different levels of granularity:** the ability to make decisions based on information obtained from all granular levels of the page structure, from columns and paragraphs to individual text fragments

At the lowest level of granularity, a set of rules or functions is used to generate confidence measures to represent the likelihood of neighbouring segments belonging to the same segment. To simplify processing, these functions can only make use of information within the respective two text blocks (such as co-ordinates, font size, textual similarity, etc.) and their proposed classification (such as paragraph, table cell, etc.)

Higher level decisions, such as the location of a paragraph or table column, are made using algorithms similar to those described earlier in this paper. However, instead of returning just one result, they return a set of possible *candidate* results. To select the best result, the scores in the corresponding lower-level blocks are evaluated. Thus the algorithm can choose the best “fit” on all levels of granularity by recursively evaluating and combining these scores.

We expect this approach to be more flexible and tolerant of small idiosyncracies in page layout that would cause problems for simpler algorithms. We also believe that, by adding additional rules, it will be relatively simple to extend the algorithm to cope with new layout structures such as tables and lists.

References

1. Baumgartner, R., Flesca, S., Gottlob, G.: Visual Web Information Extraction with Lixto. In Proc. of the 27th Intl. Conf. on Very Large Databases. (2001) 119–128
2. Lovegrove, W., Brailsford, D.: Document Analysis of PDF Files: Methods, Results and Implications. In Electronic Publishing. Vol. 8 Nos. 2–3 (1995) 207–220
3. Krüpl, B., Herzog, M., Gatterbauer, W.: Using Visual Cues for Extraction of Tabular Data from Arbitrary HTML Documents. In Proc. of the 14th Intl. World Wide Web Conf. (2005) 1000–1001
4. Russ, D., Summers, K.: Geometric Algorithms and Experiments for Automated Document Structuring. In Math. and Comp. Modelling. Vol. 26 No. 1 (1994) 55–83
5. Ha, J., Haralick, M., Phillips, I.: Recursive X-Y Cut using Bounding Boxes of Connected Components. In Proc. of the 3rd Intl. Conf. on Document Analysis and Recognition. (1995) 952–955
6. Altamura, O., Esposito, F., Malerba, D.: Transforming Paper Documents into XML Format with WISDOM++. In Intl. J. of Doc. Anal. and Recog. (2001) 4(1) 2–17
7. Kieninger, T.: Table Structure Recognition Based on Robust Block Segmentation. In Proc. of Document Recognition V. (1998) 22–32