Intelligent Text Extraction from PDF Documents*

Tamir Hassan and Robert Baumgartner
Database and Artificial Intelligence Group
Institute of Information Systems
Vienna University of Technology
Favoritenstraße 9-11, 1040 Wien, Austria
{hassan, baumgart}@dbai.tuwien.ac.at

Abstract

In recent years, PDF has become the de-facto standard for the exchange of print-oriented documents on the Web. This includes many business documents such as financial reports, newsletters and patent applications, and there are many commercial applications that require data to be extracted from these documents and processed by computer systems.

A number of products currently exist on the market that navigate, extract and transform data from HTML pages; a process known as wrapping. *One such methodology is* Lixto[1]*, a product of research at our institute. However, none of these products are currently able to work with PDF files. We are investigating this possibility as part of the NEXTWRAP project. This paper describes our work in progress, and details some of the low-level page segmentation techniques that we have investigated.*

1. Introduction

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 have little or no explicit structural information, making automated machine processing and data extraction a difficult task. HTML documents, in comparison, are inherently structured, and this structure is used by wrappers to locate instances of the data to be retrieved. Although later versions of the PDF specification support the use of XML tags to denote logical elements, these often need to be added manually, and hence are seldom found in business documents.

When a human reader views the document, various *layout conventions* indicate its logical structure. In order to make PDF files amenable to machine processing, this logical structure needs to be rediscovered from the layout, and from the 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. Our 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 the HTML parse tree, a hierarchical representation of the web page. The user can then 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. Commercial PDF Tools

It is worth mentioning the large number of PDF utilities that can be found in the marketplace today. For example, there are several products which purport to convert PDF to HTML, most of which are inexpensive commercial products. One thing that is common to these packages is that they perform little or no understanding of the document's structure.

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

[1] Lixto, `http://www.lixto.com`

Proceedings of the 2005 International Conference on Computational Intelligence for Modelling, Control and Automation, and International Confe
Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'05)
0-7695-2504-0/05 $20.00 © 2005 IEEE

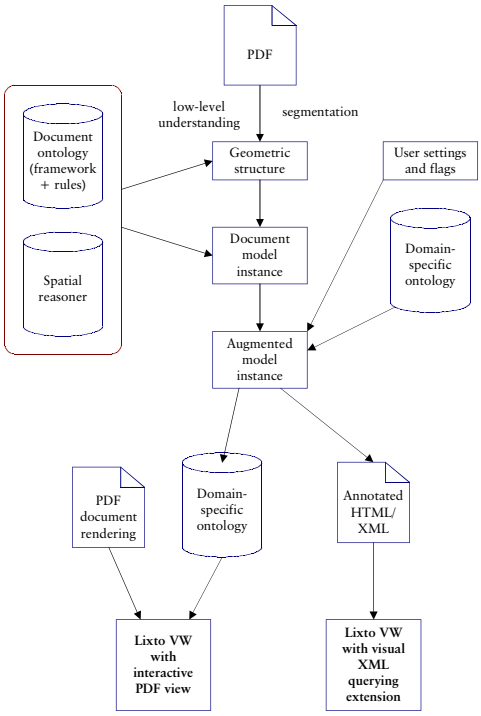


Figure 1. Outline of possible methods for wrapping from PDF

We have investigated one such package, the Advanced PDF to HTML Converter from Archisoft[2], and found that it merely reproduced the layout of the original PDF in HTML. Heuristics were used to merge text into lines, and these lines were placed on the HTML page using `<div>` elements with an absolute pixel position relative to the line's original position in the PDF. This type of conversion does not yield a HTML document that is repurposable in any way, as the problem of rediscovering the logical structure has simply been sidestepped by the conversion process.

Other tools, including Google's "View PDF as HTML" feature, were found to give similar results. Thus, we need to look into more advanced solutions for wrapping PDF documents with Lixto.

2.3. 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 [4, 5] and classification [5] that work on a binarized image as input. Whilst we could make use of these algorithms by rasterizing the PDF, we believe that this would effectively be taking a step backwards. The object information that is present in the PDF file is already at a higher level; for example, blocks of text are already classified as such. Therefore, our approach is to segment the page directly on the object data (see section 4).

There has also been significant work, particularly in the area of understanding tabular data, which deals with documents in monospaced ASCII format [3, 6]. 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 [3] and this has proved useful at several levels of the document understanding process (see section 4).

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 in-built line-finding algorithm was used to access the line objects directly. By using 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 clarify the structure 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.

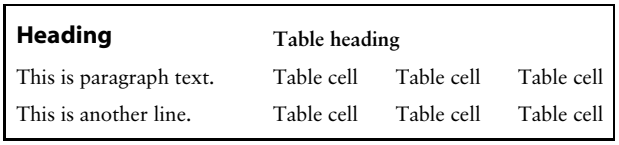
3. Overview of the Project

The first step in the extraction process is known as *layout analysis* or *segmentation*; to divide the document into blocks that can be said to constitute the smallest logical entity in the document. Typically these consist of paragraphs, headlines, titles and captions. We have experimented with some algorithms for this process, and they are described, together with preliminary results, in section 4.

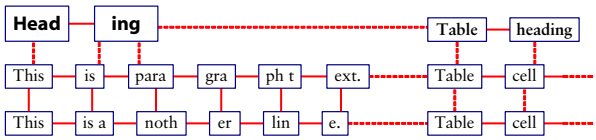
The overall process for wrapping from PDF is described in Figure 1. Once the low-level segmentation is complete, we use an ontology-based framework to store the rules and relationships between the various objects, and to determine the logical structure of the document. A spatial reasoning module is also planned that sidesteps the understanding process and allows the user to select objects based on geometric constraints. The result of this process is a multi-layer object model (see Figure 4), where each layer represents different groupings of objects on a different level of granularity.

This framework will also allow us to "plug-in" reasoning modules for specific domains, such as newspapers or journals, to obtain more specific classifications of objects. User settings will determine if and where these modules are used.

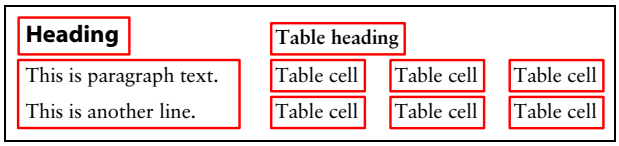
[2] `http://www.archisoftint.com/logiciels/recr_us.htm`



(a) A simple example of paragraph text and tabular content on a page



(b) Neighbourhood graph of figure 2(a). Vertices that join text fragments within the same segment are drawn with solid lines



(c) The content of figure 2(a) after segmentation

Figure 2. Neighbourhood graph example

The final stages of the process will be to integrate the new methods with the existing Lixto software. As the Lixto Visual Wrapper currently works with HTML documents, one such solution would be to serialize the document model instance into a similar tree-structured format such as annotated HTML or XML. Thus the proven techniques for wrapping from HTML could be re-used to extract data from PDF. However, such a conversion is likely to lead to some loss of data, and we therefore intend to research wrapping approaches that work directly on our PDF document model. We also intend to enable the user to interact with a visual rendition of the PDF file, with the complexity of the underlying representation being entirely hidden.

4. Page Segmentation

Segmentation methods can be divided into two general approaches: top-down and bottom-up. We have experimented with both, and have developed a prototype using the PDFBox[3] and XMIllum[4] libraries.

In our segmentation algorithms we have made use of the following techniques:

- **Neighbourhood graph:** Text fragments are stored in an undirected graph where each node represents a text fragment (or higher-level composite object) and all neighbouring objects are connected by edges. A simple example is given in Figure 2(b). This data structure makes bottom-up clustering algorithms much simpler by reducing the number of comparisons required.

- **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 division* objects. 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.

- **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 position.

  This term was introduced in [3], where a horizontal projection profile was used to determine the location of individual columns in tables. In [4] 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 also been implemented here.

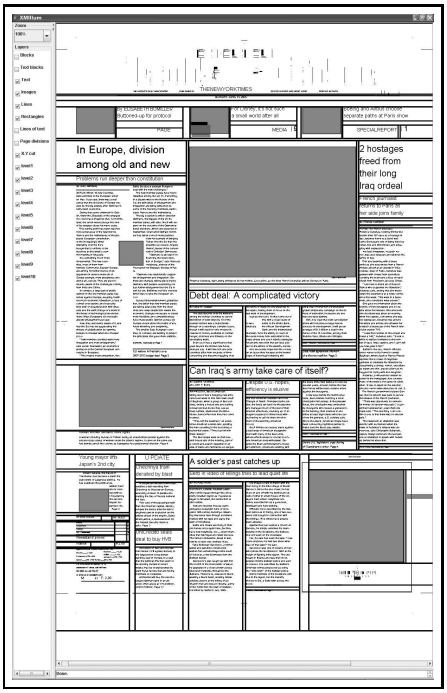
4.1. The Segmentation Process

We have implemented a top-down segmentation method based on [4]. The page is recursively divided by examining the whitespace density graph 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* is 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 3(b).

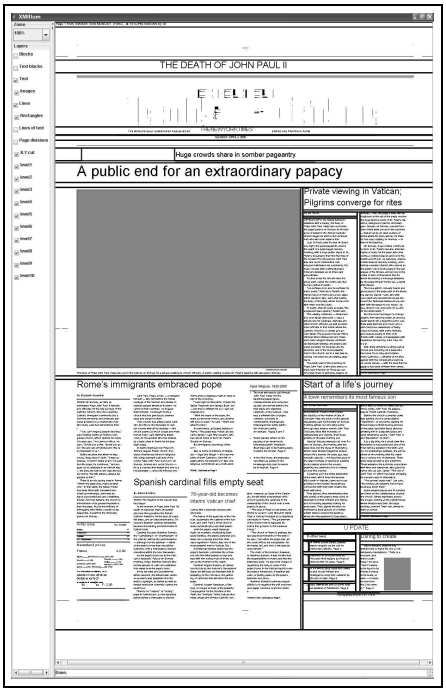
The benefit of this approach is that the resulting hierarchical structure can easily be adapted to represent the logical structure of the page by combining levels where the division or "cut" is made in the same direction.

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

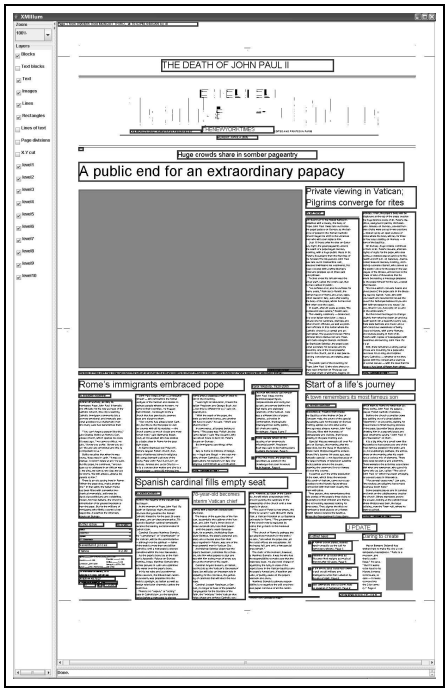
(a) The original front page of the *International Herald Tribune* newspaper



(b) Successful top-down segmentation (X-Y cut) of (a)



(c) A different issue of the *IHT*, where top-down segmentation fails



(d) The same issue as in (c), where bottom-up clustering succeeds

Figure 3. Results of various segmentation algorithms on the *International Herald Tribune* newspaper

Proceedings of the 2005 International Conference on Computational Intelligence for Modelling, Control and Automation, and International Confe
Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'05)
0-7695-2504-0/05 $20.00 © 2005 IEEE

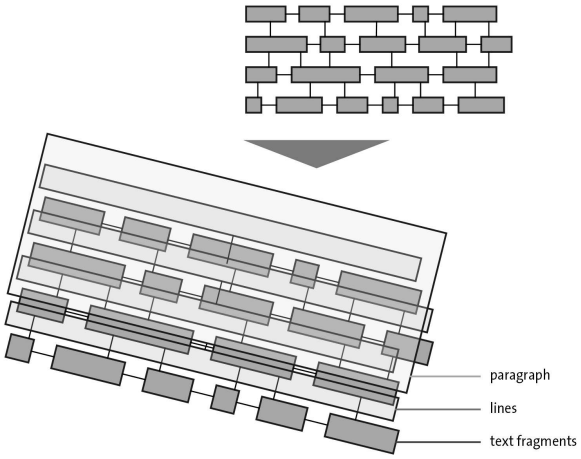


Figure 4. Our layered model

However, the top-down approach fails to completely segment certain layouts, as shown in Figure 3(c), where the entire bottom-left of the page is not segmented at all. Here one can see that it is not possible to draw a single horizontal or vertical line to divide this segment further.

We have therefore also experimented with a bottom-up clustering approach. Our algorithm iteratively examines each line in turn, from top to bottom. In a multi-column page each column is analysed separately. Lines are merged into complete paragraphs if they are below a threshold distance and share the same font size. The threshold distance is a function of the modal line spacing. The result is shown in Figure 3(d), where it can be compared to the unsuccessful result of the top-down approach (Figure 3(c)).

5. Conclusion

After investigating several segmentation algorithms we have found that both top-down and bottom-up approaches have their relative merits and disadvantages, and that the results can be improved by combining both approaches.

One important application for the Lixto suite that has not yet been fully investigated is the ability to extract data from tables on a page. When run on PDF files containing tables, the algorithms described in this paper did not give predictable results, sometimes clustering a complete column as one object, and sometimes keeping each individual cell as a single object. Our current research, as explained in the next section, is directed towards making this possible.

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

We refer again to our layered model (Figure 4), in which we represent the document in various levels of granularity. When a higher level decision is made, such as the location of a paragraph or table column, the scores in the corresponding lower-level blocks are evaluated. Thus, when the algorithm is faced with several possible higher-level groupings, it can choose the best "fit" on all levels of granularity by recursively evaluating and combining these scores.

Furthermore, our representation of the PDF as a graph has naturally led us into experimenting with graph clustering and edge-removal techniques and their application to page segmentation. Our preliminary results show that these methods are more flexible than those described earlier in this paper.

We believe this approach to be closer to how a human reader would process a document, and we hope that it can be more easily adapted to other objects, such as tables, in the future. It may also help to bridge the gap between layout analysis and document understanding, making it possible to use ontological techniques for both.

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] Russ, D., Summers, K.: Geometric Algorithms and Experiments for Automated Document Structuring. In *Mathematical and Computer Modelling.* Vol. 26 No. 1 (1994) 55–83

[4] 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

[5] Altamura, O., Esposito, F., Malerba, D.: Transforming Paper Documents into XML Format with WISDOM++. In *Intl. J. of Document Analysis and Recognition.* (2001) 4(1) 2–17

[6] Kieninger, T.: Table Structure Recognition Based on Robust Block Segmentation. In *Proc. of Document Recognition V.* (1998) 22–32