AIDAS: Incremental Logical Structure Discovery in PDF Documents

Anjo Anjewierden

Social Science Informatics, University of Amsterdam
Roetersstraat 15, 1018 WB Amsterdam, The Netherlands
anjo@swi.psy.uva.nl

Abstract

We describe the approach AIDAS *uses to extract the logical document structure from PDF documents. The approach is based on the idea that the layout structure contains cues about the logical structure and that the logical structure can be discovered incrementally.*

1. Introduction

AIDAS is part of a research project in which the aim is to turn technical manuals into a database of indexed training material. The role AIDAS plays in this project is to take a PDF file, extract the logical structure and assign indexes to each element in this logical structure. The indexes can either be about the content of the element (e.g. "this section is about the rear part of a car", "this is a schematic drawing of the control unit") or about how the element can be used in instruction (e.g. "this section provides a brief overview of the components of a car"). An instructor can retrieve appropriate training material by querying the database using the indexes.

The advantages of an automated approach to indexing training material and making it available electronically may be apparent. Whereas until recently the industrial partners in the project literally used scissors and glue to produce the material, they can now use the indexed database and cut-and-paste to presentation programs.

In order for AIDAS to be able to do its work we distinguish the following stages:

Interpreting PDF.
The starting point for AIDAS is a document in PDF [1]. PDF (Portable Document Format) is a page description language that is powerful in terms of the rendering capabilities it provides and widely used for document exchange. Section 2 describes how AIDAS extracts the layout structure from PDF.

Discovering the logical document structure.
The fragments that AIDAS needs to store correspond to the logical document structure (sections, tables, images, items, etc.). During this stage the layout structure is *incrementally* analysed in order to discover the logical structure (see Sections 3–4).

Indexing and fragmenting the logical structure.
The logical structure produced by the previous stage is annotated using a domain ontology and reasoning about the logical structure [4]. During this stage AIDAS looks at the content of the logical structure, for example comparing the title of a section to the list of concepts in the ontology. If a match between a title and a concept occurs, the section is indexed as a fragment with the name of the concept as the topic. This stage is not further discussed in this paper.

Storing the document in a multi-media database.
This takes the annotated fragments and converts them to a database. Text fragments are stored as ASCII and graphical fragments are stored as SVG [8].

2. PDF vs. scanned images

PDF represents a document as a set of instructions. When these instructions are processed by the PDF image model a bitmap is generated which can be rendered on a graphics device. The instructions in PDF fall into four categories: (1) *control* instructions which work on the image model and produce no output; (2) *text* instructions render glyphs; (3) *graphics* instructions render lines, curves and rectangles etc.; and (4) *image* instructions render bitmapped images.

Many document analysis systems take a scanned image as the source and extract text, graphics and images from it. Such systems could also deal with PDF by including a pre-processing step that renders PDF in a bitmap. We have chosen for a design in which the stream of PDF instructions

0-7695-1263-1/01/$10.00 © 2001 IEEE

is converted to a set of text, graphics and image objects. This conversion is implemented on top of xpdf [5].

The *layout structure* used by AIDAS consists of the set of objects generated by the conversion. There are about ten different classes of layout objects (text, line, rectangle, image, curve, etc.). And each of these classes has about ten features (position, font face, font size, color, line width etc.). A complication of PDF that does not occur with scanned images is that there are an infinite number of different PDF documents that render precisely the same image. For example, a word can be a single text object or there can be separate text objects for each character in the word. However, the layout structure generated from PDF is more reliable than extracting the layout structure from scanned images. There are complications, however, for example the issue of visibility. A PDF document could contain an instruction to render some text followed by an instruction to render a filled rectangle on top of it.

3. Logical structure discovery

The *logical structure* (sections, item lists, tables, etc.) is not explicitly available in the layout structure and needs to be discovered. Various approaches to logical structure discovery have been suggested. If the document style (e.g. font face and size of section titles, line spacing, number of columns) is available then the use of top-down grammars that incorporate the document style is possible, but even then extensive error recovery is necessary to cater for idiosyncrasies [3]. An alternative approach is to first discover a hierarchy of layout objects (glyph, word, line, text block, column) and then map this hierarchy on the logical document structure hierarchy [9]. This approach will work well if the two hierarchies can be easily mapped on each other.

The approach in AIDAS is based on the idea that layout objects represent only their layout features explicitly, but that these features contain cues about the role in the logical structure [6], [7]. For example, a text object in a large bold font (the *form*) contains the cue that it could be a section title (the *function*), and a text object containing the literal * (form) could be a bullet (function). The distinction between form and function is also seen at the logical structure. For example, a paragraph (form) can be the body of a bullet or an item (function) or it can, along with other paragraphs, be the body of a section. An extreme case is that the paragraph may be the title of a section in the case of a title spanning multiple lines.

AIDAS uses this idea by assigning a set of possible functions to each layout object and incrementally chunking them to more complex objects. This process is performed incrementally until the logical structure is produced. Determining the possible functions of a layout object can be done in a bottom-up fashion, whereas the function is determined using top-down techniques (grammars).

4. Implementation

The first step performed by AIDAS is to identify the overall layout of a document. This step identifies the columns, headers and footers, and the dominant font. All of these are determined by statistical analysis and can be overridden by the user. This process is easier in PDF compared to scanned images.

Thereafter AIDAS breaks up each page in *segments*. Drawings are recognised by many graphical elements versus few text elements and tables are recognised by a lot of text classified as floating. All remaining segments are text segments.

4.1. Classifying individual layout objects

The next step is to look at individual layout objects generated by the PDF interpreter. During this step AIDAS classifies each layout object on several dimensions: geometry, markup and (textual) content.

For example, given the following layout object:

```
<text>
  <layout x=25 y=200 w=180 h=14
          face="Times-Bold" size=12/>
    1.1 Introduction
</text>
```

the classification results in:

```
<text>
  <layout x=25 y=200 w=20 h=14
          face="Times-Bold" size=12/>
  <geometry alignment=left/>
  <markup size=larger emphasis=bold/>
  <function sectionnum="1.1"/>
    1.1
</text>

<text>
  <layout x=50 y=200 w=155 h=14
          face="Times-Bold" size=12/>
  <geometry indentation=25/>
  <markup size=larger emphasis=bold/>
    Introduction
</text>
```

The *geometry* element has been abstracted from the column boundaries, the *markup* element from the dominant font and the *function* element has been determined by matching the text against the patterns for section numbers.

This form abstraction can proceed without regard of the surrounding text because no information is lost. Should it later be discovered that the "1.1" object is not a section number then the abstractions are ignored and, perhaps, the two text objects are concatenated again (for example as part of the phrase "see also Section 1.1 Introduction").

Other examples of abstractions are bullet characters ("*", "-", etc.), enumeration indicators ("(a)", "1)", etc.) and, specifically for technical manuals, references to figures ("General Theory of Operation (Fig. 14-15A)".

4.2. Shallow grammars to detect logical elements

The logical structure itself is discovered by taking the set of layout objects, sorting them on y/x position and then running a set of shallow grammars on them. Each of these grammars detects a specific element of the logical structure and leaves the objects that do not match untouched. This architecture has several pleasant characteristics: (a) the grammars are shallow and complexity is roughly linear; (b) new grammars that discover other logical structure elements can easily be added; and (c) no error recovery is necessary as the grammars ignore unrecognised input.

A simplified version of the grammar for detecting section titles is given below:

```
sectiontitle(SectionTitle) -->
  section_number(Num),
  section_name(Name),
  SectionTitle is
    sectiontitle(sectionnumber(Num),
                 sectionname(Name)).

sectionnumber(Num) -->
  Num is next_node,
  match(Num, function, sectionnum=N),
  match(Num, geometry, alignment=left),
  match(Num, markup, emphasis=bold),
  match(Num, markup, size=larger).

sectionname(Name) -->
  Name is next_node,
  match(Name, geometry, indentation=N),
  match(Name, markup, emphasis=bold),
  match(Name, markup, size=larger).
```

A match results in the replacement of layout objects by a structural object that represents a section title:

```
<sectiontitle>
  <sectionnumber>
    <text>
      <layout x=25 y=200 w=20 h=14 .../>
      <geometry alignment=left/>
      <markup size=larger ... />
      <function sectionnum="1.1"/>
        1.1
    </text>
  </sectionnumber>
  <sectionname>
    <text>
      <layout x=50 y=200 w=155 h=14 .../>
      <geometry indentation=25/>
      <markup size=larger .../>
        Introduction
    </text>
  </sectionname>
</sectiontitle>
```

The incremental nature of the process can be illustrated further by considering how the next level in the logical structure, sections, is discovered:

```
section(Section) -->
  section_title(Title),
  section_body(Body),
  Section is
    section(Title, sectionbody(Body)).

section_title(Title) -->
  Title is next_node,
  element(Title, sectiontitle).

section_body(Body) -->
  /* Collect everything until a
     sectiontitle element is seen. */
```

In the current implementation grammars for the following logical elements are used: section titles, sections, bullets, bullet lists, items, item lists, headings, paragraphs, tables and figures.

The final step is to remove all abstractions from the objects and output the logical structure itself. For the above example this results in:

```
<section level=2 id="1.1">
   <sectiontitle>Introduction</sectiontitle>
   <sectionbody>     ...       </sectionbody>
</section>
```

This logical structure is then processed by the indexing and fragmentation module of AIDAS [2].

5. Experimental results

At the time of writing AIDAS is being used by the industrial partners to fragment technical manuals in three domains: car repair; military equipment; and electronic control systems for traffic lights.

The car repair manuals have a very specific logical structure. For each topic (e.g. body work) there is a separate section with short descriptions and lots of figures of the car and how to use tools for repair. A domain specific shallow grammar has been added to recognise this structure and it performs very well according to the users.

The manuals for the military equipment (radar-based interception) consist of a mix of theory and technical drawings. The theoretical material is in four columns (A3 pages) and the drawings are in separate books with captions. One of the problems here was to create a link between the books containing the theoretical sections and the books containing the technical drawings. We have not received a formal evaluation for this material, but experiments on parts show that extracting the logical structure from the text is more than acceptable.

One of the virtues of the incremental approach used in AIDAS is that it is "robust" with respect to unexpected input. Although it will not recognise the unexpected input (and therefore leave it as layout objects), domain specific shallow grammars can be added to recognise additional logical structures if necessary. This is very important for technical manuals, as there is no such thing as "a logical document structure" for technical manuals as is illustrated by the military and car domains. Technical manuals take into account the skills of the readers: in the military domain theory is supplemented by figures, whereas in the car repair domain the emphasis is on manual skills which is mainly supported by figures.

The figure on the last page shows a screendump of the tool after analysing a page from the military domain. The rectangles represent segments. To the left of each recognised logical structure element is a the name of the element (e.g. sectiontitle, item, paragraph, heading). Colour is used to show to the user what part of the text is recognised.

6. Acknowledgements

The work presented in this paper has been supported by the European Commission as part of the IMAT (Integrating Materials and Training) project. We gratefully acknowledge comments on previous versions of this paper by Jan Wielemaker and Robert de Hoog. We also would like to thank the three anonymous reviewers whose comments we have tried to take into account.

References

[1] Adobe Systems Incorporated. *PDF Reference version 1.3*. Addison Wesley, Boston, second edition, July 2000.

[2] A. Anjewierden and S. Kabel. Automatic indexing of PDF documents with ontologies. 2001 (submitted).

[3] T. Hu and R. Ingold. A mixed approach toward an efficient logical structure recognition from document images. *Electronic Publishing*, 6(4), December 1993.

[4] S. Kabel, B. Wielinga, and R. de Hoog. Ontologies for indexing technical manuals for instruction. In *Workshop on Ontologies for Intelligent Educational Systems*, Le Mans, July 1999.

[5] D. Noonburg. xpdf: A C++ library for accessing PDF. www.foolabs.com/xpdf.

[6] K. Summers. Toward a taxonomy of logical document structures. In *Electronic Publishing and the Information Superhighway: Proceedings of the Dartmouth Institute for Advanced Graduate Studies (DAGS'95)*, pages 124–133, 1995.

[7] K. Summers. *Automatic discovery of logical document structure*. PhD thesis, Cornell University, August 1998.

[8] W3C. SVG (Scalable Vector Graphics). www.w3c.org.

[9] Y. Wang, I. Phillips, and R. Haralick. From image to SGML/XML representation: one method. In *International workshop on Document Layout Interpretation and its Applications (DLIA 1999)*, Bangalore, India, September 1999.

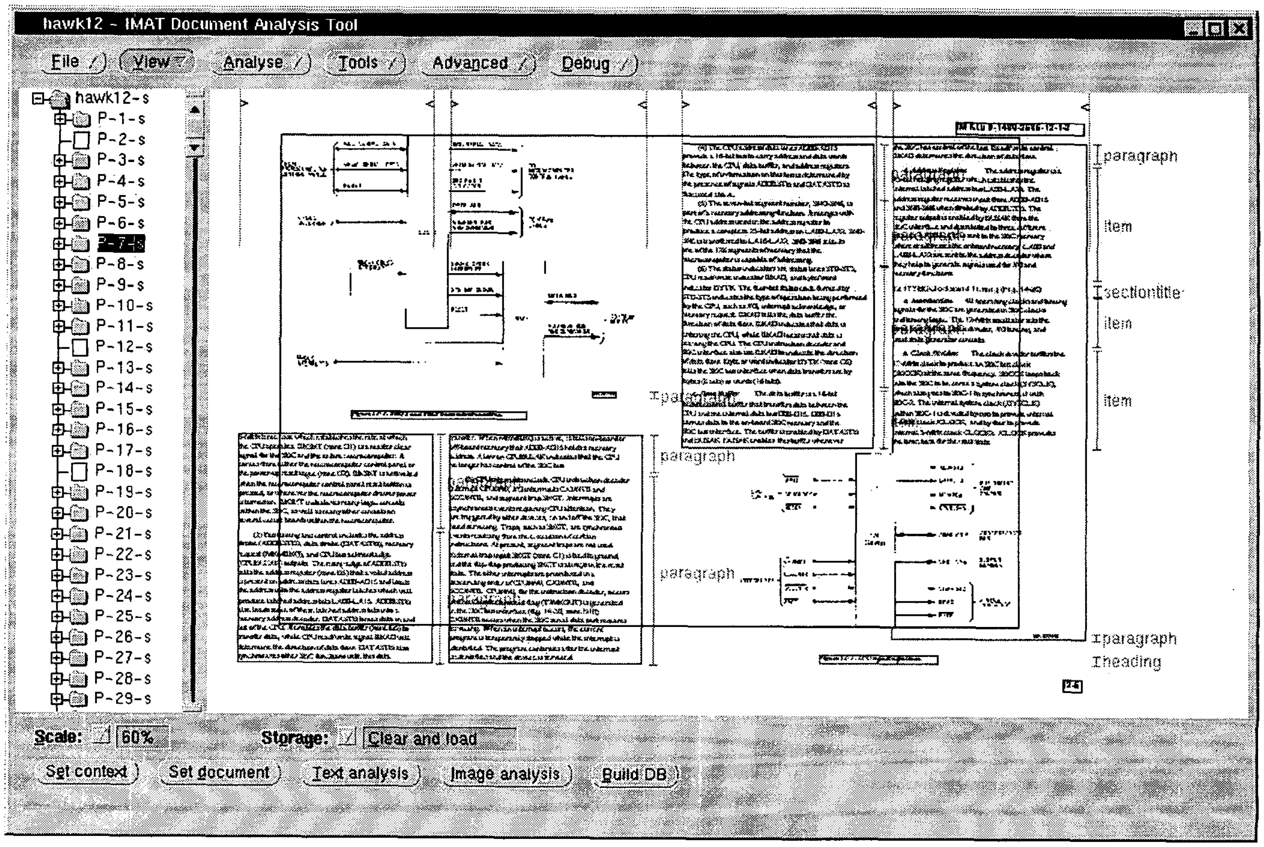

Figure 1. AIDAS analysing a technical manual