

# Automatic indexing of documents with ontologies

Anjo Anjewierden and Suzanne Kabel

Social Science Informatics, University of Amsterdam  
Roetersstraat 15, 1018 WB Amsterdam, The Netherlands  
email: \{anjo, kabel\}@swi.psy.uva.nl

## Abstract

Indexing large bodies of data is necessary to enable satisfactory search results. Ontologies serve as fixed vocabularies to index data from different viewpoints. We describe how AIDAS, a software tool, automatically divides the source data (PDF documents) into reusable chunks, how it automatically indexes these chunks and stores them in a database to enable reuse.

## 1 Introduction

A large body of knowledge is available as formatted text written with a particular purpose in mind. One example of such bodies of text are technical manuals. These manuals contain all information about a particular system or device, such as a helicopter, as viewed from the manufacturer (reference and maintenance). Reusing these manuals for *instructional* purposes requires a different perspective on the content of the material and often also a more attractive presentation.

In this paper we describe our approach to partially automating the process of indexing technical manuals such that an author of training material can retrieve the appropriate text- and picture fragments. Subsequently, these fragments can be reused in presentation programs and other course material.

A technical manual of a helicopter consists of thousands of pages with text, pictures and technical drawings. Domain specific terms, for example *rotor blades*, will occur many times and a simple keyword based search will result in many hits each of which has to be checked manually. For example, if one would like to introduce the rotor blades, one might want to search for “a general description” and a picture, to motivate the students. Supporting such queries requires that fragments are not only indexed on the words they contain, but also on: (1) domain concepts they refer to; (2) semantic aspects like the kind of description (structural, procedural); (3) the scope (general, elaborate); (4) physical aspects (text, picture, schema); and (5) instructional aspects (kind of knowledge, role in reuse).

Automated *document analysis* combined with *ontology indexing* techniques is the approach taken in the IMAT project (Integrating Manuals and Training) [8] in which the

aim is to reuse technical manuals as training material. We have developed the necessary ontologies and implemented the approach in a software tool called AIDAS.

The following stages are distinguished in the document analysis and ontology based indexing process. The source material is taken to be available 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. The fragments that need to be stored and indexed correspond to the logical document structure (sections, tables, images, items, etc.). Fragmentation is performed automatically using document analysis techniques as described in Section 2. The next step is to index the text- and picture fragments. The logical structure is analysed and various techniques are used to define a mapping between a concept in one of the ontologies and fragments in the document. Each mapping found becomes an index on the fragment (Section 3).

The approach has been used in industrial settings. In the conclusions (Section 4) we review this practical experience.

## 2 Document Analysis

Traditionally, document analysis is the art of taking a scanned document (including OCR) and discovering the logical document structure. Thereafter the logical structure is processed in an application dependent manner. Document analysis itself mainly considers geometry (e.g. alignment), markup (e.g. use of fonts), whereas particular applications will also take into account the meaning of the text or image.

Document analysis clearly falls within the realm of AI: it is a non-trivial task which humans can do rather well. Although it may appear that the relevance of document analysis has decreased with electronic publications (also on the internet), this is not the case. High-quality documents are exchanged in formats in which the logical structure is no longer present, for example PDF (see Section 2.1).

Considerable progress has been made in document analysis over the last decade. In general, one can state that when the structure of the document is known in advance various systems will perform very well. In IMAT we are dealing with the very broad category of technical manuals, rather than documents which have a familiar structure (e.g. scientific articles). This complicates matters enormously as the “document style” is not known in advance and has to be discovered itself. AIDAS applies a wealth of AI and knowledge-based methods to achieve sensible results.

### 2.1 PDF

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.

We have developed a conversion from the instructions to *layout* objects represented in XML. This conversion is based on XPDF [7]. 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, colour, line width etc.).

## 2.2 Logical structure discovery

Logical structure discovery can be viewed as converting a set of layout objects to a single hierarchical logical structure object. The set of layout objects represents the physical structure of the document and the logical structure represents how the document is organised.

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 [5].

The approach in AIDAS [2] is based on the idea that layout objects not only represent their layout features explicitly, but that these features contain cues about the role in the logical structure [9]. For example, a text object in a large bold font could play the role of a section title, and a text object containing the literal \* could play the role of a bullet. AIDAS uses this idea by assigning a set of possible roles to each layout object and incrementally chunking them to more complex objects. This process is performed incrementally until the logical structure is produced.

## 2.3 Implementation

A document analysis system has to reason about (1) *geometry*, i.e. the positioning of objects on a page; (2) *markup*, fonts and styles; and (3) sometimes *lexical* information, e.g. to recognise that (a) is the start of an item. Reasoning about the meaning is not normally necessary to discover the logical structure, but is obviously necessary for indexing (Section 3).

The representation of a layout object is directly generated from the PDF interpreter:

<b>I(Content, Area, Subs, Abstractions)</b>	Layout objects
<i>Content</i> is a term describing the object content itself, for example <code>text(String, Font)</code> is some string and <code>image(File, Width, Height)</code> is a bitmap image. <i>Area</i> is the bounding box of the object. <i>Subs</i> is a set of sub-objects, for example a path can consist of several line segments. <i>Abstractions</i> is a set of abstractions about the object.	

The first step AIDAS performs is to *classify* all layout objects. The classification method considers each object and adds abstractions based on rules that are dependent on global features of the document (dominant font, column margins, etc.). One set of rules contains abstractions on the geometry and another set of rules defines the abstractions based on markup. For example:

### Initial (PDF) representation

```
l(text('12.1', 'Times-Bold-12'), area(60,160,34,14), [], []).
```

### After geometry and markup abstractions

```
l(text('12.1', 'Times-Bold-12'), area(60,160,34,14), [],
  [ position=left      /* aligned to left margin */
  , column=3          /* in the third column */
  , fontsize=large    /* larger than default font */
  , emphasis=bold     /* emphasis */
  ]).
```

The next step is to determine whether a layout object contains text that can indicate elements of the logical structure. This can be determined by an ontology that maps concepts about the logical document structure onto the tokens in the text. This ontology contains typical patterns that indicate, for example, bullets (\*, -, (\*)) and section numbers ('1.1.', '1-1').

A classification method compares all layout objects to the patterns and inserts abstractions when a match occurs. The abstractions are not considered as precise classifications as the context also has to be taken into account. For example “see section 1.1 Introduction” contains “1.1” which matches a rule for a section-number, but it is not a section number part of the logical document structure.

The representation of a structure object is modelled after the representation of XML:

```
s(Element, Attributes, Body, Abstractions)           Structure objects
  Element is the name of the element, Attributes is the set of attributes for the element
  and Body is the content of the element. For example <p align=center>Hello
  is represented as: s(p, [align=center], ['Hello'], []).
```

The final step is to take order into account. AIDAS does this by defining a grammar for all possible elements of the logical structure. The grammar for detecting a section head from layout objects and generating a structure object is:

```
sec_head(s(sectionhead, [number=Section], [Title], [])) -->
  section_number(Section),
  section_title(Title).

section_number(Section) -->
  [ l(text(Section,_), _, _, Abs) ],
  { has_abstraction(Abs, section_number, true) },
  { has_abstraction(Abs, position, left) },
  { has_abstraction(Abs, fontsize, or(default,large)) },
  { has_abstraction(Abs, emphasis, or(bold,italic)) }.

section_title(Title) -->
  [ l(text(Title,_), _, _, Abs) ],
  { has_abstraction(Abs, position, indented) },
  { has_abstraction(Abs, fontsize, or(default,large)) },
  { has_abstraction(Abs, emphasis, or(bold,italic)) }.
```

The grammars *constrain* the relevance of the layout abstractions before a conclusion is drawn. For example, as the above grammar shows, a layout object can only play the

role of a section number if it looks like a section number, is aligned to the left of a column, is at least in the default font, has emphasis; **and** is followed by a layout object that has the necessary features to be a section title.

In conclusion, AIDAS uses classification to find relevant features of layout objects and grammars to find out whether the features are relevant enough to emit an element in the logical structure.

### 3 Using ontologies to index the logical structure

A fragmented document can be indexed according to different points of view. We have developed ontologies corresponding to the following viewpoints: general and syntactical (fragment ontology), semantic (description ontology), instructional and domain [6].

We view a fragment (text, picture) as consisting of a content and a set of indexes. The indexes correspond to nodes in the ontologies and the user can specify any combination of keys to retrieve the fragments based on the indexes.

There are different approaches to index fragments. Some index values are derived automatically from the source, others need to be deduced using mappings between ontologies and the content of the fragments, combined with heuristics, while still other value types can not be determined up-front and need to be added to fragments with human intervention.

#### 3.1 Fragment ontology

In the fragment ontology, general aspects are captured, as well as information about medium and type of representation and structure. Examples of general attributes are *size* for text fragments and *width* and *height* for picture fragments. *Structural type* represents the form and visualization of a fragment. *Representational type* stands for the original source representation of the information. Examples of attribute values for medium, structural and representational type are:

medium	structural type	representational type
text format	annotation	pictorial representation
ascii	box	analog representation
pdf	document segment	basic pictorial elements
rtf	equation	structured representation
html	headings	chart

#### 3.2 Domain ontology

A domain ontology is a conceptual description of a domain, e.g. a certain system or device. Titles in technical manuals are often similar to concepts in the domain ontology. For example when a section describes how a particular component works or how it can be replaced, the title will at least include the name of the component. AIDAS uses a procedure that takes a title, normalises it to a set of relevant words and then compares this list of words with the concepts in the ontology. Although this process may seem trivial, a lot of knowledge about language is required. If the title of a fragment or the caption of a

figure corresponds to a concept, then the topic of the fragment becomes the name of the concept and recursively all sub-fragments will also get this topic. This takes advantage of the fact that in technical manuals the logical structure often corresponds to the part-of structure in the domain ontology.

The domain ontology should at least contain abbreviations and the most important synonyms of components. AIDAS supports the creation of domain ontologies from for example parts lists, table of contents, or from scratch.

### 3.3 Description ontology

Semantic aspects of fragments are represented in a description ontology, in which description type and scope are captured. *Description type* reflects the view from which a fragment was written, e.g. a *structural description*, refined into *description of components*, *description of location of components*, and so on.

A way to derive semantic values is to map the words in a fragment to a set of terms that indicate a high probability of a certain description type. For example, a fragment is indexed as a structural description if contains terms like *component* and *consists of*.

Description scope provides information on the nature of the fragment, e.g. *general*, *short* or *elaborate*. The description scope of a fragment can often be found by considering the relative position of the fragment. A simple rule is that if a chapter has a certain topic and the first section of the chapter is called *Introduction* or *General* then we can index the section on the topic with a description scope of short.

Another way to derive semantic values such as the description scope, is by looking at the occurrences of a certain topic in a larger context. For example, the value of the slot description scope can be derived by checking whether two fragments have CWAR-radar as the topic. If this is the case, the first is likely to have a general or short description scope, while the second probably has an elaborate description scope. Another general rule is that the deeper the nesting the more detailed the content will be.

### 3.4 Instructional ontology

Instructional aspects (see [3], [4]) are grasped in an instructional ontology, with a focus on maintenance training. An instructional description of a fragment can consist of a *learning goal*, *knowledge type*, *instructional strategy*, *instructional activity*, *instructor and learner actions* and *material use*, each with their own set of possible values. Learning goal describes the desired outcome of an instructional curriculum, e.g. *learning the learner to perform inspection task X*. Knowledge type stands for the kind of knowledge that is necessary for the learner to master the learning goal, like *terminological knowledge* or *procedural knowledge*. Instructional strategy refers to the way of teaching, for example *coaching* or *articulation*. Instructional activities in maintenance training are for instance *setting goals*, or *presenting tasks*. These are broken down into *instructor* and *learner actions*, like *present the learning goal*, *comprehend the learning goal*, and *give assignments*, *comprehend and do assignments*. Material use points out what will be the role of the fragment in the context of lesson-material. For example, a fragment will be used in the training material as *learning goal*, *assignment*, *example*, *explanation* or *illustration*.

Instructional values are role-based values. The role a fragment can play in a new context is hard to derive automatically from its original context. However, it can be done beforehand by determining what fragment types are typically suitable for this new context. A mapping to suitable fragments is made via description type, which *is* derived automatically. In other words: given this learning goal, knowledge type and instructional strategy, fragments with structural descriptions are most suitable.

Manual indexing is performed during the creation of training material or training scenarios, or in general during reusing the fragment in its new context. This way, there is no “real” indexing effort, and while creating training material the database with fully indexed fragments is built up.

## 4 Conclusions

In the IMAT project, three prototypes of AIDAS, combined with a retrieval tool, have been developed and used. Three industrial user partners developed training material with these prototypes and used it in the classroom. The evaluations revealed possible improvements with respect to fragment size and indexing terminology of the subsequent prototype. Besides other evaluations that concern the usability of the tool, and the comprehensibility of the ontologies, these evaluations shed light on the output of the AIDAS tool.

The results presented in this paper can possibly be applied to other areas. The two major dependencies are: the use of PDF and the focus on technical manuals. In general, we believe that the combination of document analysis and the use of ontologies which provide various viewpoints on documents is very powerful.

In principle, PDF can be replaced by any representation for which it is possible to convert the input to layout objects with a bounding box (see Section 2.3). In an other context we have developed a pre-processor that turns plain text documents (i.e. email messages) into the layout representation. Unfortunately, representations such as HTML only provide a very minimal set of layout primitives which, moreover, people tend to “misuse”. An initial prototype of AIDAS was based on HTML as input, but the amount of layout information lost by converting documents to HTML is so large that many layout features could not be discovered consistently.

Using AIDAS in contexts different from technical manuals can be achieved by extending the logical document structure to be discovered. For example, in the context of scientific papers the document structure needs to be extended with rules for detecting bibliographic entries and abstracts.

The work presented in this paper has been supported by the European Commission as part of the IMAT (Integrating Materials and Training) project. We would like to thank the anonymous reviewers for their comments.

## References

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

- [2] A. Anjewierden. AIDAS: Incremental logical structure discovery in PDF documents. In *6th International Conference on Document Analysis and Recognition (ICDAR)*, pages 374–378, Seattle, September 2001.
- [3] H. Chen. A methodology for characterizing computer-based learning environments. *Instructional Science*, 23:183–220, 1995.
- [4] R.M. Gagné, L.J. Briggs, and W.W. Wager. *Principles of Instructional Design*. Harcourt Brace Jovanovich College, Fort Worth, fourth edition, 1992.
- [5] T. Hu and R. Ingold. A mixed approach toward an efficient logical structure recognition from document images. *Electronic Publishing*, 6(4), December 1993.
- [6] 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.
- [7] D. Noonburg. xpdf: A C++ library for accessing PDF. [www.foolabs.com/xpdf](http://www.foolabs.com/xpdf).
- [8] Y. Barnard R. de Hoog and B. Wielinga. Imat: Re-using multi-media electronic technical documentation for training. In *EMMSEC Conference*, Stockholm, June 1999.
- [9] 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)*, pages 124–133, 1995.