IJDAR (2002) 5: 1–16
Digital Object Identifier (DOI) 10.1007/s10032-002-0080-x

International Journal on Document Analysis and Recognition IJDAR

Document understanding for a broad class of documents

Marco Aiello[1,2], Christof Monz[2], Leon Todoran[1], Marcel Worring[1,⋆]

[1] Intelligent Sensory Information Systems, University of Amsterdam, Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
[2] Institute for Logic, Language and Computation, University of Amsterdam, Plantage Muidergracht 24, 1018 TV Amsterdam, The Netherlands
e-mail: {aiellom,christof,todoran,worring}@science.uva.nl; http://www.science.uua.nl/∼aiellom,christof,todoran,worring

Received: March 15, 2001 / Revised version: March 18, 2002

Abstract. We present a document analysis system able to assign logical labels and extract the reading order in a broad set of documents. All information sources, from geometric features and spatial relations to the textual features and content are employed in the analysis. To deal effectively with these information sources, we define a document representation general and flexible enough to represent complex documents. To handle such a broad document class, it uses generic document knowledge only, which is identified explicitly. The proposed system integrates components based on computer vision, artificial intelligence, and natural language processing techniques. The system is fully implemented and experimental results on heterogeneous collections of documents for each component and for the entire system are presented.

Keywords: Document understanding – Logical object classification – Reading order detection – Qualitative spatial reasoning – Natural language processing

1 Introduction

Document analysis can be viewed as reversing the process of document authoring. It is therefore important to consider the choices an author makes. In the creation process, the author starts with a rough idea about the content. The author then structures his/her thoughts by considering the logical organization of the material, e.g., dividing the material in chapters and deciding on the intended reading order. When the final digital document is printed on paper, the underlying logical structure of the document is obscured by the actual layout conventions. The author has, however, the possibility to encode some of the logical information using layout typesetting conventions, e.g., by using a specific font size, style and arrangement on the page. Therefore, the layout structure of a printed document carries, besides the artistic message, some information about the logical structure. In fact, for simple documents, a human reader can determine the logical structure from layout and typesetting information only. In complex documents, the human reader completes his/her understanding of the document structure by reading the text. Therefore, to reverse the authoring process for document analysis purposes, both structure and content are essential.

Generic logical structure information, needed in document understanding, is called *document knowledge*. The knowledge is encoded as a set of layout and typesetting conventions for a document class.

The importance of document knowledge for document image analysis was emphasized by various authors [24, 19, 10, 27, 20, 26]. Based on generalization/specialization hierarchies the knowledge is organized into two, three, or more levels. In [10], Cesarini et al. distinguish only between two types of knowledge: generic and specific. Nagy et al. in [24] define three levels of knowledge: generic knowledge, class-specific knowledge, and publication-specific knowledge. For some applications more detailed organization of document knowledge can be considered. For instance publication-specific knowledge can have a subclass called title-page knowledge. A nice, though brief, survey of the type of knowledge used in prominent systems is [19].

As in this paper we focus on the analysis of heterogeneous collections of documents based on generic knowledge, we adopt a basic distinction of document knowledge into two classes: document-generic knowledge and document-specific knowledge. The *document-specific knowledge* is the document knowledge that is specific for a narrow class of documents. The *document-generic knowledge* is the document knowledge common to a broad variety of documents. An example of document-specific knowledge is that in ACM transactions the running text is Times Roman, with a font size of 10 points. An example of generic knowledge is that, usually, the title is written in a larger font size than the paragraph text.

Obviously, a system will be more effective in document understanding, when it uses more knowledge about

⋆ All authors contributed equally to this paper

the document class. The issue is to differentiate between specific and generic knowledge. A system that has hard-coded all knowledge about a given class of documents will understand that class of documents only. A robust system able to process a broad class of document should first exploit the generic knowledge to assign the document to a certain class [32], and then use specific knowledge.

Besides the generic/specific dichotomy, another aspect has to be considered in classifying document understanding systems. This is the *complexity* of documents that can be handled.

Though the term document complexity is often used in the literature, there is no standard formal definition for it. In the context of this paper, we will refer to *document complexity* in terms of the number of document objects on the document page. For example, for reading order detection, the more textual objects a document has, the more complex it is. Recently, more refined definitions of document complexity based on different document analysis tasks have been proposed [36].

Document understanding systems presented in the literature can be roughly classified into four classes of increasing complexity:

- systems processing simple documents of one specific document class [11,12,19];
- systems able to process simple documents from broad classes [38,27,20];
- systems processing complex documents of one specific document class [37,26];
- systems able to process a broad class of complex documents [3,35,15].

Note that systems able to process complex documents are a superset of systems processing only simple documents. However, the systems able to process a broad class of documents are not necessarily a superset of systems processing specific classes, as they might not achieve the same performance as specifically designed systems.

We are interested in the systems able to process a broad class of complex documents. The design of a system able to process *any* class of documents remains still a challenge. We believe that the latter type of systems should:

- use a document representation able to capture the most complex situations;
- use all generic knowledge available;
  - geometric information: positioning, spatial relations, global information;
  - content information: textual features, lexical information.

The remainder of the paper is organized as follows: in the next section we consider document analysis systems in more detail, and we relate them with the system we present here. In Sect. 3, we describe the document representation model. In Sect. 4, we present our document understanding method in depth. Performed experiments are presented in Sect. 5, and discussed in Sect. 6.

2 Related work

Systems developed for specific limited domain classes, such as mail automation, form processing, and processing of business letters [11,12] report good results. They implicitly or explicitly use specific knowledge. In [11], Cesarini et al. present a tax-form processing system that uses specific knowledge and interaction with the user. The document representation is based on attributed relational graphs, which allow an accurate and flexible representation of the form class.

Lee [19] describes a systems that analyzes journals from IEEE, specifically TPAMI. The specific knowledge is hardcoded in rules and threshold values. Therefore it is hard to adapt their system to other document classes. They, however, report good results for processing documents from journals of the IEEE, other than TPAMI, as they have the same geometric characteristics.

Walischewski [38] presents a system to process business letters. Given class specific knowledge and a training set, the system can adapt itself to process new document classes. The document representation is based on an attributed directed graph and spatial relations which are appropriate for processing complex generic documents. No textual information is used.

The use of learning modules to extract the set of rules to map layout into logical structure from the document knowledge leads to more adaptable systems as those presented by Sainz and Dimitriadis [27] and Li and Ng [20]. Again, these systems ignore the important role of textual content.

Other systems process more complex, multi-article documents such as newspapers. A prominent example is the system developed by Tsujimoto and Asada [37]; which is tuned to process regular multi-column black-and-white papers. Both for layout and logical structure detection, domain knowledge is hard-coded into four transformation rules. The use of a tree based representation for the document restricts the class of documents that can be processed. In addition, the rules work only for the specific document class and cannot be adapted to other classes.

In [26], Niyogi presents a system that explicitly uses both specific and generic knowledge to process newspapers. Textual content is not taken into account. The document representation is based on XY-trees [25] which limits the complexity of the documents that can be handled by the system.

Klink [15] uses explicitly both specific and generic knowledge to process different classes of documents exemplified by business letters and technical papers. Textual features and geometrical relations are considered in the classification process. The classification module is based on fuzzy-matched rules. The fuzziness allows small variations, making the system a bit generic, thus the rules are domain-dependent. For each document class, specific rules are considered. In addition, the system is limited in the variety of documents it can analyze by the use of a tree representation for the documents. The reading order detection problem is not addressed.

In this paper, we make a step towards generality. First, we develop a graph-based document model similar to the one proposed in [11], but we extend it to represent more complex document structures. Unlike other document models, the model is not restricted to a regular layout, and it allows for the representation of overlapping document objects. Second, we design a document understanding system that uses the generic knowledge on both layout and textual content. The document understanding system builds on techniques from computer vision, artificial intelligence, and natural language processing.

From computer vision we borrow statistical decision trees based on geometric feature vectors for the classification of textual blocks into logical objects. Similar statistical methods using decision trees or neural networks on feature vectors were used in [39,40,18,31]. Even though the techniques used are similar, the final goal is different. Rather than being interested in the distinction between text and non-text for every document object, we are interested in classifying all the textual document objects into different categories, such as title, footnote, and page number.

From artificial intelligence we borrow notions from the field of qualitative reasoning and constraint satisfaction. In particular, we consider bidimensional extensions of Allen's interval relations [4], that is, rectangular relations. To the best of our knowledge, bidimensional Allen relations have been used in document image analysis in three cases [15,33,38]. In all these approaches, bidimensional Allen relations are used as geometric feature descriptors, at times as labels for graphs and at other times as layout relations among document objects. Thus, the use of Allen relations is relegated to feature comparison, and it is not used for performing any other kind of reasoning, as in the present case.

From the field of natural language processing we borrow the use of statistical methods such as bi-grams and tri-grams. To the best of our knowledge, such techniques have not been used before for reading order detection.

Comparing the whole document understanding system proposed here with others, the main difference lies in the step we take towards a complete detection of the logical structure. We integrate the logical labeling and the reading order detection modules into a document understanding system which can work on heterogeneous classes of documents. Assuming the layout structure as given, we detect the logical labels and the reading order.

3 Document description

The goal of the system presented here is to process a broad class of documents. To this end, we introduce a document model generic and flexible enough to capture the most complex document structures. We consider descriptors for the content and structure of the document based on the document model.

3.1 The document model

Models for document representation need at least two distinct structures: one for the layout information, to encode the presentation of the document, and one for the logical information, organizing the content into related logical entities. Usually, models allow for only one layout structure per document page, and one logical structure per document. Complex documents, however, require several views on both the layout and logical information.

In the model we propose, a document $\mathcal{D}$ is a set $\mathcal{G}$ of layout or geometric structures, and a set $\mathcal{L}$ of logical structures:

$$\mathcal{D} = \langle \mathcal{G}, \mathcal{L} \rangle.$$

The set of layout structures $\mathcal{G}$ is a collection of views $\mathcal{G} = \{g^1, ..., g^n\}$. The layout structure g^i is a set $\mathcal{O}_g^i$ of geometric document objects, and a set $\mathcal{R}_g^i$ of geometric relations among them:

$$g^i = \langle \mathcal{O}_g^i, \mathcal{R}_g^i \rangle.$$

Hence, each type of geometric relation is represented as a graph. The vertices are the document objects, and an edge represents the existence, and possibly the value, of a named relation between the document objects. This graph can be a tree for a simple relation, but in general, it is a directed graph.

The set of logical structures $\mathcal{L}$ is defined in a similar way as a collection of views: $\mathcal{L} = \{l^1, ..., l^m\}$. A logical structure l^i is a set $\mathcal{O}_l^i$ of logical document objects and a set ${\mathcal{R}_l}^i$ of logical relations:

$$l^i = \langle \mathcal{O}_l^i, {\mathcal{R}_l}^i \rangle.$$

The set of logical document objects ${\mathcal{O}_l}^i$ in view i holds the content of the logical elements of the document and the meaning represented as a logical label. The logical structures are represented as weighted graphs.

3.2 Document object description

Document objects are the basic elements used in the document representation. As defined in the document model, the document objects are part of both the geometric and the logical structures.

For the purpose of this paper, we consider picture and text document objects. Pictures are only considered as a whole. Different granularities for text elements are considered, such as characters, words or text-blocks. Here, we take text-blocks as the smallest entities carrying a logical meaning.

The shape of the document objects is assumed to be rectangular. For non-rectangular shapes, we simply consider the bounding box.

A description of the layout using content information and the positioning of document objects on a page, i.e., bounding boxes, is sufficient to reproduce the original document image, and, for instance, to print it on paper. Therefore, to describe a geometric document object, two main categories of features are used:

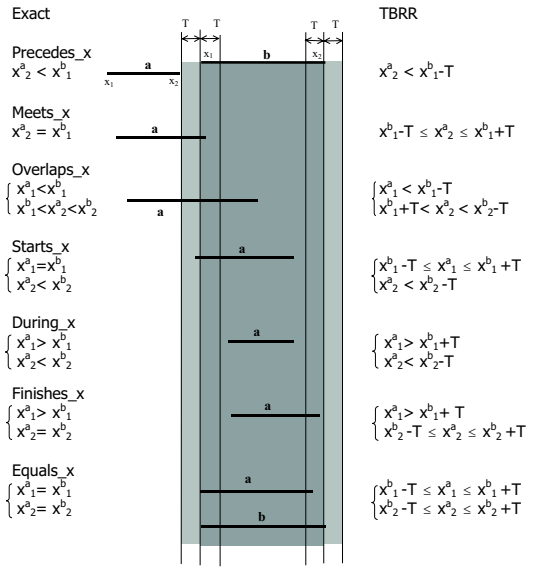


Fig. 1. The definitions of exact (*left*) and TBRR (*right*) rectangle model relations shown for the X coordinate. In the middle is an illustration. For simplicity, only the direct relations are illustrated; the inverse relations are obvious. The interval b is 'fixed', while the interval a is 'sliding' from left to right showing the possible relative positions. The borders of size T are drawn at the endpoints of b

– geometric features;
– content features.

For each geometric document-object a set of features is determined. This *feature vector* is selected such that it describes, as accurately as possible, the instantiation of a document in a class.

To describe the content of documents in a broad document class, generic features are required. The features are:

– the *aspect ratio* - defined as the ratio between width and height of the bounding box;
– the *area ratio* - defined as the ratio between block area and the page area;
– the *font size ratio* - defined as the block font size divided by the most frequently used font size in the document;
– the *font style* - defined as an enumerated type, with possible values: regular, bold, italic, underline;
– the *content size* - defined as the number of characters;
– the *number of lines*.

This feature vector will be completed with another feature, based on the neighborhood relation, which is defined in the following section.

For a large heterogeneous collection of documents there is a limited number of logical document objects common to all of them [23]. Therefore, only the following textual document objects are considered as logical labels: `Title, Body, Caption` and `Page Number`.

3.3 Layout structure description

Besides the layout description presented above, which is needed for reproduction of the document image, explicit geometric relations are needed for understanding the layout. These geometric relations fall into two major classes:

– global arrangement of objects on the page;
– spatial relations between objects on the page.

For representation of spatial relations between objects represented by their bounding box, we use qualitative relations. First, consider the extension of Allen's interval relations [4], originally devised for 1-D temporal intervals, to two dimensions [22,38,8]. On both the X and the Y axis the thirteen relations: *precedes, meets, overlaps, starts, during, finishes, equals*, and their inverses are considered. Abbreviations of these interval relations used in the rest of the paper are *p, m, o, s, d, f, e, pi, mi, oi, si, di, fi*, where the last six are the inverses of the first six, respectively.

Due to the inherent inaccuracy in document image analysis, relations based on exact coincidence of points are of little use. Thus, we make a shift in the interpretation of Allen's relations. Instead of considering two interval extremes to be equal if they share the same coordinates, we consider them equal if they are closer than a fixed distance T. This can be dually seen as if the bounding boxes have a *thick boundary*. We name the set of 13 Allen's relations thus interpreted *Thick Boundary Rectangle Relations* (TBRR).

The thickness of the boundary is the same for all objects in the document and it is fixed with respect to the page size. The best values for T are found through experimentation and are usually in a range of 1–3% of the average block size of the page, [2]. There is an additional constraint on the value of T with respect to the size of the smallest document object: it should not exceed half the size of the shortest side of the smallest bounding box. Referring to Fig. 1, one sees how the TBRR relations are more tolerant in the establishment of a relation between two intervals. For example, interval a meets interval b not only if $x_2^a = x_1^b$, but also if $x_1^b - T \leq x_2^a \leq x_1^b + T$.

With the TBRR interpretation Allen's relations maintain the jointly exhaustive and pairwise disjoint property; meaning that, given two rectangles, one and only one TBRR relation holds among the two rectangles.

To prove the jointly exhaustive (JE) property in the above statement, one has to show that the union of the thirteen TBRR relations is the whole $\mathbb{R}$ line space. To prove the pairwise disjoint (PD) property, one has to show that the intersection of any two given sets of the thirteen is empty.

In Fig. 2, the set of points (*1, 2, 3, 4, 5*) where the endpoints of any given interval can be situated are shown. The sets *2* and *4* are closed, the others are open. In Table 1, all possible relations achievable when positioning the endpoints of a in the allowed sets are reported.

Consider the left endpoint of a as situated in *1*. The possible positions for the right endpoint of a are in *1, 2, 3, 4, 5*. If x_2^a is also located in *1* then $\infty < x_1^a < x_1^b - 3T$

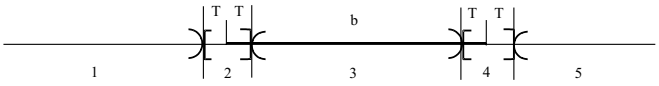


Fig. 2. The possible positions for the endpoints of the sliding interval a relative to the fixed interval b

Table 1. The possible relative interval position for the endpoints of the a interval

x_1^a	x_2^a	*Relation*
1	1	p
	2	m
	3	o
	4	fi
	5	di
2	3	s
	4	e
	5	si
3	3	d
	4	f
	5	oi
4	5	mi
5	5	pi
$\mathbb{R}$	$\mathbb{R}$	JE ($\cup$)
ϕ	ϕ	PD ($\cap$)

and $x_1^a + 2T < x_2^a < x_1^b - T$ hold. Writing down all these intervals one can easily see that the union of all the intervals where x_2^a can be situated, is the part of $\mathbb{R}$ situated to the right of $x_1^a + 2T$. Because x_1^a is sliding from left-to-right all of $\mathbb{R}$ is covered. Therefore, the TBRR are jointly exhaustive. In a similar way, considering x_1^a in all five possible sets, one shows that the intervals where x_2^a can be situated are disjoint.

The TBRR is a set of qualitative relations representing the spatial relations of the document objects on the page. For every pair of document objects o_1 and o_2, one X and one Y interval relation hold. If one considers the pair in reversed order, the inverse interval relation holds. Therefore the directed graph g^i representing these relations is complete.

For global arrangement of document objects on the page, we use a neighborhood relation. Two document objects o_1 and o_2 are considered *neighbors* if they share an edge in the Voronoi diagram [6]. The Voronoi diagram is computed for the centers of gravity of the bounding boxes representing the document objects. This relation is stored in a weighted graph where the nodes are the document objects. An edge represents the existence of the neighborhood relation between them. The weight of the edge is the actual Euclidean distance between them.

This neighborhood relation is added as a Boolean value to the feature vector describing textual document objects presented in previous section. This feature indicates whether the current text document objects is neighbor to a figure or not.

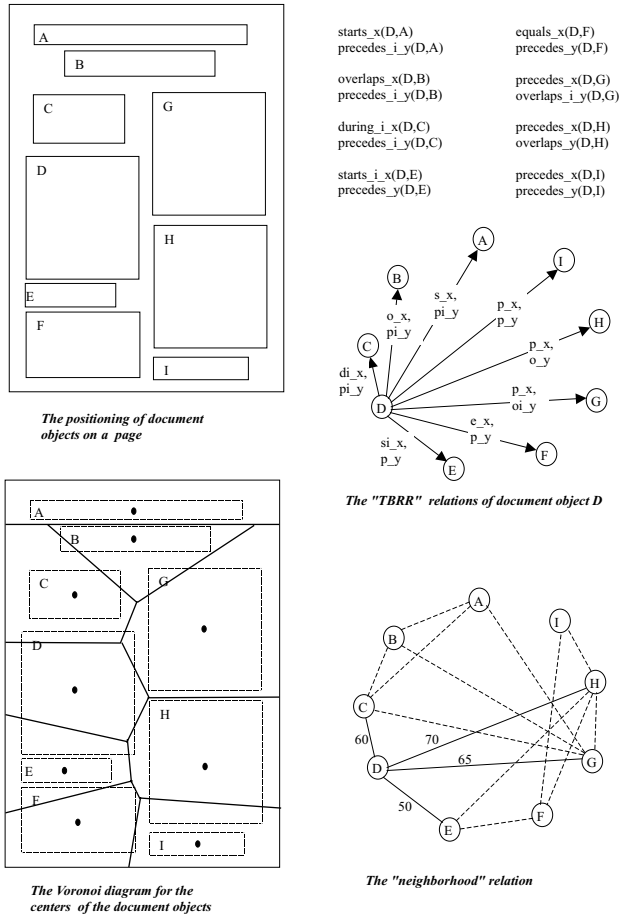


Fig. 3. Exemplification of the layout relations. The upper part shows the TBRR relations for all pairs where document object **D** is the first element of the pair. The bottom part shows the Voronoi diagram, and the neighborhood relation for all pairs with starting element D (*solid*) and other (*dashed*)

An example of interval and neighborhood relations is shown in Fig. 3.

The proposed layout representation captures the essential information for a given document, which is required by our system for further logical analysis.

3.4 Logical structure description

As for logical relations between document objects, we consider a partial ordering relation *BeforeInReading* which holds for two document objects if one is to be read before the other one. Note that this does not mean that it has to be read *immediately* before, but just that it is before in the reading order. This partial ordering can be extended to a total ordering among document objects which is the *reading order*.

Recall that only *Body, Title, Page Number* and *Caption* are permissible logical labels. *Page Numbers* are relevant for identifying pages of the document. They are not important for the reading order on a page. *Captions* and their associated pictures are considered as auxiliary

to the reading of the document. After reading a caption, the main linear reading order continues. Therefore, the reading order is considered only among *Body* and *Title* document objects.

In this paper, we make the assumption that for a document page there is only one reading order. This is a limitation of our current system, as there are examples (e.g., newspapers) where on a single page there are many independent articles, which can be read independently of one another in any order.

4 Document understanding

Document analysis starts with the document image and ends with its complete logical structure. For this purpose two main steps are needed: one to extract the layout structure, called *layout detection*; and another one to determine the logical structure, called *document understanding*. As document authoring is a non-reversible process, document knowledge is essential in the document analysis process.

The document knowledge is mainly used in document understanding. There are two main phases in document understanding. In the first phase, the layout document objects are grouped and classified as logical objects. Then, among logical document objects, logical relations are determined. In this paper, we consider the reading order as a logical relation. The whole process is sketched in Fig. 4.

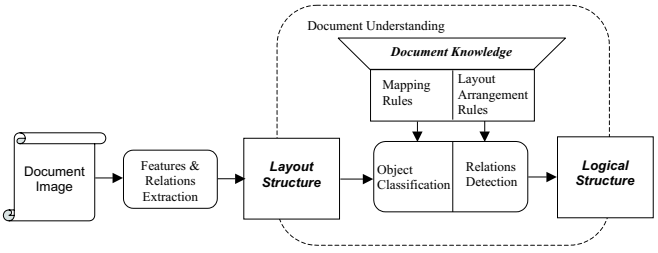


Fig. 4. The main steps of document analysis, and the role of document knowledge

4.1 Document knowledge

As we aim to process a broad collection of documents we consider only generic knowledge. The document-generic knowledge used can be grouped in two main categories:

1. Content mapping rules from logical to layout document objects:
   (a) titles are written in larger than average font size and/or in a non-regular style;
   (b) title objects usually consist of less than three lines;
   (c) a paragraph is written with a uniform regular font size and style;
   (d) the page number is located on the top or the bottom of the page;
   (e) the caption is a neighbor of a picture document object.
2. Layout arrangement rules reflecting the logical structure:
   (a) in Western culture the reading order is from left-to-right and from top-to-bottom;
   (b) the reading order is consistently either column- or row-oriented;
   (c) paragraphs start with a capital letter, and end with a termination mark such as a period.

Even though there might be some exotic documents that do not obey the above rules, they form the basis for our document understanding system.

4.1.1 Knowledge acquisition There is a question about the acquisition of the general knowledge necessary for the document analysis tasks. In this paper, the origin of the knowledge comes from common-sense reasoning and statistical methods.

In particular, for the task of logical labeling, the generic knowledge synthesized in the content mapping rules from logical to layout document objects is obtained by common-sense reasoning and by verification using statistical measurements on documents coming from datasets. First, we made common-sense hypotheses based on experience and observations. Second, we studied the histograms of feature vector values computed on documents of given datasets. Measurements confirmed the initial observations.

In addition, in the case of the spatial reasoning module common-sense reasoning is the origin of knowledge. For instance, it is common knowledge in Western culture that documents are read from top-to-bottom and from left-to-right. However, there are other ways to acquire document-related knowledge. In [2], the notion of a *document encoding rule* is proposed. Such rules capture how the layout of a document conveys information and the intent of the author to the reader of a document. The rules, that can be expressed in various formalisms such as in a LaTeX class file, can come from different sources. Rules can be obtained by common-sense reasoning, they can be obtained directly by the author of the document class (e.g., the printer of a journal disclosing the LaTeX class file he/she uses), and they can be obtained via learning techniques (e.g., the first-order learned rules described in [5]).

Finally, the natural language processing module uses n-grams to compute the transition probabilities between two text blocks. To build a general model, 400,000 journal abstracts were part-of-speech tagged and the distribution of bi- and tri-grams was extracted. This sort of knowledge is domain independent and falls under the category of generic document knowledge.

4.2 Logical object classification

For every textual document object, the geometric features described in Sect. 3.2 are combined into a feature

vector. At this point each document object is represented by a set of seven features as follows:

- the *aspect ratio* - defined as the ratio between width and height of the bounding box;
- the *area ratio* - defined as the ratio between block area and the page area;
- the *font size ratio* - defined as the block font size divided by the font size most frequently used in the document;
- the *font style* - defined as an enumerated type, with possible values: regular, bold, italic, underline;
- the *content size* - defined as the number of characters;
- the *number of lines*;
- the *neighbor to figure* - expressed as a yes/no feature.

The feature vector consists of continuous and enumerated values. We should, therefore, select a method able to deal with both continuous and enumerated values, [29].

Based on these features we assign one of the logical labels *Caption, Body, Title* or *PageNumber* to a document object. The problem of assigning a label to each document object is a standard statistical pattern recognition problem [14]. Therefore, we use standard tools.

4.3 Reading order detection

As defined in Sect. 3, the reading order is determined through the intermediate logical relation *BeforeInReading*. The possible reading orders are detected independently for document object type *Body* and *Title*, respectively. Then, these reading orders are combined using a Title-Body connection rule. This rule connects one *Title* with the left-most top-most *Body* object, situated below the *Title*.

The reading order is determined by deploying both geometrical and content information. For this purpose, we build:

- a spatial reasoning (SpaRe) module based on the spatial relations TBRR;
- a Natural Language Processing (NLP) module based on lexical analysis.

Due to the generality of the document knowledge used, it is likely that one gets more than one reading order, especially for complex documents with many text blocks.

The two graphs of *BeforeInReading* relations are combined to extract the final reading order relation.

As described in Sect. 3, the relations defined yield a directed graph. Using the assumption that only one reading order is present on a page, the reading order is a full path in this graph. As the graph is cyclic, standard *topological sort* [16] for finding a full path in the graph cannot be used. We use instead a modified version of it (cf., Appendix B in [2]). To be precise, the nodes in the graph are sorted by the number of outgoing arrows of BeforeInReading relations. All nodes have to be reached once and only once.

The two modules built are applied in a strict order. First, the spatial reasoning module identifies a number of spatially admissible reading orders; second, the natural language processing module identifies the linguistically most probable among those. It is important to notice that switching the order of the two components is inefficient. In fact, in our system the spatial reasoning module works independently of the results of the natural processing module. The natural language component can take advantage of the pruning operated by the spatial reasoning module. Rather than considering all factorial combinations of textual document objects, the natural language module only looks at the reading orders provided by the spatial reasoning module. The advantage resides in the fact that the algorithms behind the spatial reasoner have a polynomial complexity in the number of document objects.

4.3.1 SpaRe: spatial reasoning module In the spatial arrangement approach, the document knowledge (2a) and (2b) and the layout 2D interval relations are used as a set of constraints to determine the *BeforeInReading* logical relation [3]. These constraints are satisfied using the Eclipse environment [13]. The knowledge (2a) is encoded in the rule depicted in Fig. 5 top. This states that the document object A is in BeforeInReading relation with the document object B, if either of the following TBRR relations $p_x(A,B)$, $p_y(A,B)$, $m_x(A,B)$, $m_y(A,B)$, $o_x(A,B)$ or $o_y(A,B)$ holds. In other words, this encodes the fact that documents are read from top-to-bottom and from left-to-right (note that one can go first left-to-right or first top-to-bottom) allowing for the overlapping of document objects.

Furthermore, we exploit document knowledge (2b) by considering two possible reading orders for every document:

- *row-wise*: text-blocks are read in left-to-right rows, which are then read from top-to-bottom;
- *column-wise*: text-blocks are read in top-to-bottom columns, which are then read from left-to-right.

The rules for each reading direction are presented in Fig. 5. Basically, the two directional rules, are both adding one specific constraint to the basic rule depicted in Fig. 5 top. In the row-wise reading order the diagonal direction "left-bottom to top-right" cannot be present among the BeforeInReading relations allowed. This is implemented by specifying all the other BeforeInReading relations that can occur, and leaving the non-allowed ones out. Similarly, in the column-wise reading order the diagonal direction "right-top to bottom-left" cannot be present among the BeforeInReading relations allowed.

In Fig. 6, the extracted logical relations BeforeInReading and the final reading order are shown for the document of Fig. 3. The BeforeInReading relations presented in the graph are computed by the SpaRe module. From this two possible reading orders are detected. The correct one is shown on the right side.

We remark on the polynomial complexity of this component with respect to the number of document objects. This comes from the fact that the specific constraint

```
if(   precedes_x(A,B)    OR
      precedes_y(A,B)    OR
      meets_x(A,B)       OR
      meets_y(A,B)       OR
      overlaps_x(A,B)    OR
      overlaps_y(A,B)
      )
then
      beforeInReading(A,B)
```

```
% VERTICAL COLUMNS %

if(    precedes_X(A, B)  OR
       meets_X(A, B)     OR
       (overlaps_X(A, B) AND ( precedes_Y(A,B)   OR
                               meets_Y(A,B)      OR
                               overlaps_Y(A,B) ))
                         OR
       ((precedes_Y(A, B) OR  meets_Y(A,B) OR overlaps_Y(A,B)) AND
                               (precedes_X(A,B)  OR
                               meets_X(A,B)      OR
                               overlaps_X(A,B)   OR
                               starts_X(A,B)     OR
                               finishesi_X(A,B)  OR
                               equals_X(A,B)     OR
                               during_X(A,B)     OR
                               duringi_X(A,B)    OR
                               finishes_X(A,B)   OR
                               startsi_X(A,B)    OR
                               overlapsi_X(A,B)))
then
       beforeInReading(A,B)
```

```
% HORIZONTAL COLUMNS %

if(    precedes_Y(A, B)  OR
       meets_Y(A, B)     OR
       (overlaps_Y(A, B) AND
                               (precedes_X(A,B)  OR
                               meets_X(A,B)      OR
                               overlaps_X(A,B)) )
                         OR
       ((precedes_X(A, B) OR meets_X(A,B) OR overlaps_X(A,B)) AND
                               (precedes_Y(A,B)  OR
                               meets_Y(A,B)      OR
                               overlaps_Y(A,B)   OR
                               starts_Y(A,B)     OR
                               finishesi_Y(A,B)  OR
                               equals_Y(A,B)     OR
                               during_Y(A,B)     OR
                               duringi_Y(A,B)    OR
                               finishes_Y(A,B)   OR
                               startsi_Y(A,B)    OR
                               overlapsi_Y(A,B)) )
then
       beforeInReading(A,B)
```

Fig. 5. The BeforeInReading rules encoding the spatial reasoning constraints. On top, the most generic BeforeInReading rule is displayed. The rules constraining column-wise and row-wise document understanding are displayed in the center and bottom, respectively. The latter two are the rules actually used in experimentation

satisfaction problem with bidimensional Allen relations has polynomial complexity [8], this check is operated quadratically many times (all pairs of document objects are checked), and finally a graph of BeforeInReading relations is sorted in quadratic time [2]. In short, the worst case complexity of the component is polynomial in the number of document objects present in the page under analysis.

4.3.2 NLP: natural language processing module To extract the reading order from the textual content, natural language processing is required. As we aim at generic analysis, we use shallow NLP tools, like taggers, which can contribute to the resolution of reading order ambiguities.

A tagger assigns a part-of-speech tag, such as `DT` (determiner: *the, a*), `VBD` (past tense verb: *took, said*), `SENT` (sentence boundary: . ! ?), etc. to each word or punctuation sign. When this is done successfully, document knowledge (2c) can be employed.

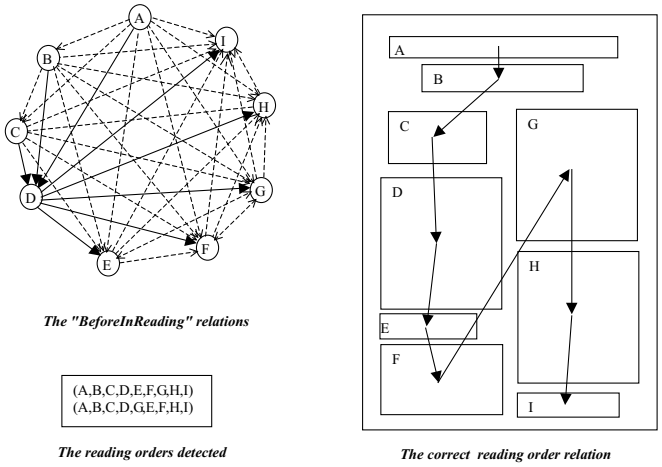


Fig. 6. Exemplification of the logical relations. The BeforeInReading relations are computed by the SpaRe module. Here the relation involving element D are drawn solid. From these two reading orders were detected. The correct one is shown on the right side

For the determination of the BeforeInReading relation, we use a probabilistic approach. Let o and o' be two document objects, we want to determine the probability $P(o\, o')$ i.e., the probability that o and o' obey the BeforeInReading relation.

First, both objects are tagged and the last two tagged words ending and the first tagged word beginning objects o and o' are identified. We refer to the last two tags of an object o by t_o^{-2} and t_o^{-1} and to the first tag by t_o^1. w_o^{-2}, w_o^{-1}, and w_o^1, refer to the words themselves.

The restriction to sequences of length 3 is mainly due to the sparse data problem, where it can be expected that most training corpora are too small to assign reliable frequencies to infrequent sequences.

The likelihood of a tagged sequence is defined as:

$$P(t^1_{o'}|t^{-2}_o t^{-1}_o) = \frac{P(t^{-2}_o t^{-1}_o t^1_{o'})}{P(t^{-2}_o t^{-1}_o)}$$

$P(t_o^{-2}t_o^{-1}t_{o'}^1)$ is computed by dividing the number of occurrences of the tag sequence $t_o^{-2}t_o^{-1}t_{o'}^1$ in a pre-tagged training corpus by the number of all trigrams occurrences. $P(t_o^{-2}t_o^{-1})$ is computed analogously. The reader is referred to [21] for a comprehensive introduction to statistical natural language modeling.

To decide the whole reading order of a document page with the textual objects $o^1 \dots o^n$ the probability is computed as the product of the transition probabilities of all consecutive text objects:

$$P(o^1, \dots, o^n) = \prod_{i=1}^{n-1} P(o^i,\ o^{i+1})$$

By computing probabilities it is also possible to rank the different reading orders. The reading order with the highest probability is taken as the genuine one.

In general, $n(n-1)$ transitional probabilities have to be computed, where n is the number of text blocks on

a page. In order to identify the order with the highest probability, all permutations of the n text blocks have to be considered, resulting in factorial complexity $O(n!)$. This is the most general case, where no further (spatial) constraints are used. In the current implementation the number of permutations is dramatically reduced by the spatial reasoning component. In our experiments, maximally four permutations had to be considered.

5 Experiments

Experiments have been performed using two collections of documents: the University of Washington UW-II English journal database [28], and the MTDB [30] from the University of Oulu, Finland. The first is a collection of 623 pages coming from conference proceedings and scientific journals. The second data set consists of 171 pages of scanned documents of various types: technical journals, newspapers, magazines, and commercial ads. Both datasets come with ground truth (GT) at the block level granularity. There is no explicit separation between the layout and the logical structure. Every document object has a layout label and a logical label. The reading order is also present in the ground truth. In the case of the MTDB there is no ground truth for the textual content and font information, thus the TextBridge OCR package [34] from ScanSoft was used.

The UW-II is a standard collection widely used in the field of document understanding, while MTDB provides a more heterogeneous set of documents. The block granularity is lower in UW-II, therefore more blocks are present per page than in MTDB. In MTDB there are also newspaper pages with multicolumn organization, while in UW-II there are scientific papers only with at most two columns.

For evaluation purposes, the documents in the data set were split into three main groups, based on their complexity:

- *trivial* documents consisting of at most three textual document objects;
- *regular* documents with the number of document objects between four and eight;
- *complex* documents with more than eight document objects.

Out of 624 document pages of UW-II 163 are of type *trivial*, 276 of type *regular*, and 185 are of type *complex*. As for MTDB, out of 171 document pages 98 are of type *trivial*, 66 of type *regular*, and seven are of type *complex*. The total number of textual document objects is 5,364 for UW-II and 894 for MTDB.

Each of the three modules of the system, i.e., the logical object classification (LoC) module, the spatial reasoning module (SpaRe) and the natural language processing (NLP) module, was tested independently. The data flow used for evaluation of the modules is shown in Fig. 7.

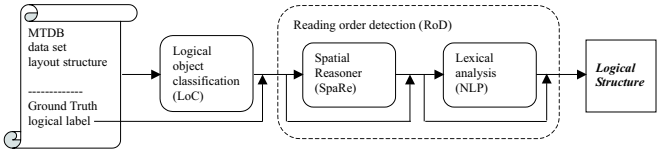


Fig. 7. The data flow in the various experiments considered

5.1 Evaluation criteria

To evaluate the individual modules and the system as a whole, we resort to standard information retrieval measures [7]: precision and recall.

The LoC module assigns a label to an object and is evaluated by considering the number of misclassifications. The precision and recall are defined for every logical type $i = \{Body, Caption, Title, Page\ number\}$ by:

$$prec_i = \frac{|C_i \cap GT_i|}{|C_i|} \qquad rec_i = \frac{|C_i \cap GT_i|}{|GT_i|}$$

where C_i represents the set of document objects classified by the LoC module as logical objects of type i and GT_i represents the set of document objects of logical type i conform the ground truth. Both precision and recall have values in the range [0, 1]. The precision $prec_i$ gets value 1 if all document objects classified as type i by the LoC module are actually of type i in the ground truth. Consequently, $prec_i$ gets value 0 if none of the document objects classified as type i are actually of type i in the ground truth. The recall rec_i gets value 1 if *all* document objects of type i in the ground-truth, are classified correctly. Consequently rec_i gets value 0 if none of the document objects of type i is classified as type i.

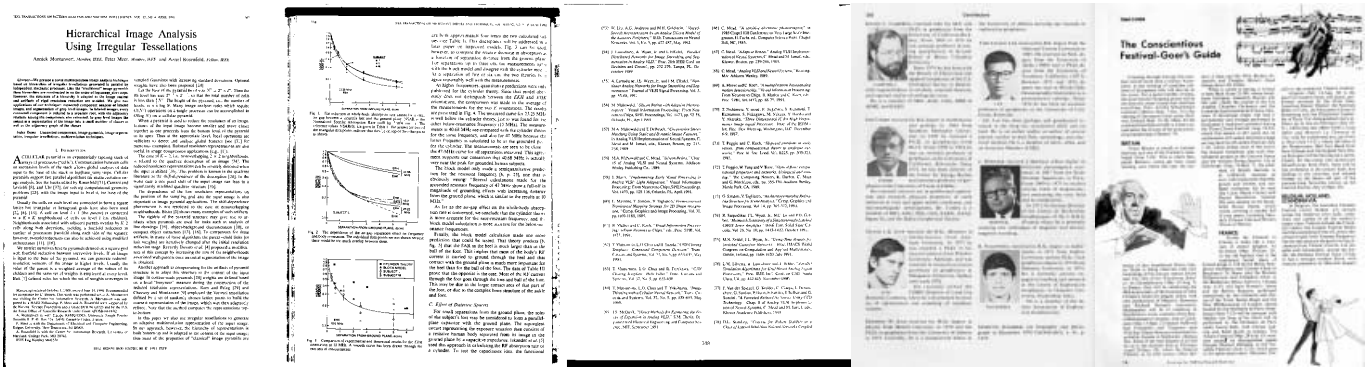
The average precision and recall computed from the four individual measures are the overall performance measures for the LoC module.

The goal of the SpaRe and NLP modules is to find among all possible orders of the blocks the single one that corresponds to the reading order.

The set of reading orders detected (D) is compared to the ground truth. The UW-II provides one unique reading order in the ground truth for all pages. On the contrary, for the MTDB dataset the ground truth defines independent reading orders on non-intersecting subsets of the textual objects within the same document; e.g., a page containing two different articles or independent text blocks with information about the authors of an article, as exemplified in Fig. 8.d. For the MTDB data collection, we consider a reading order correct if it is identical to at least one permutation of the independent reading orders as defined in the ground truth. We refer to the set of permutations of the ground truth as the set of correct reading orders (CRO). Then, the precision and recall are defined as follows:

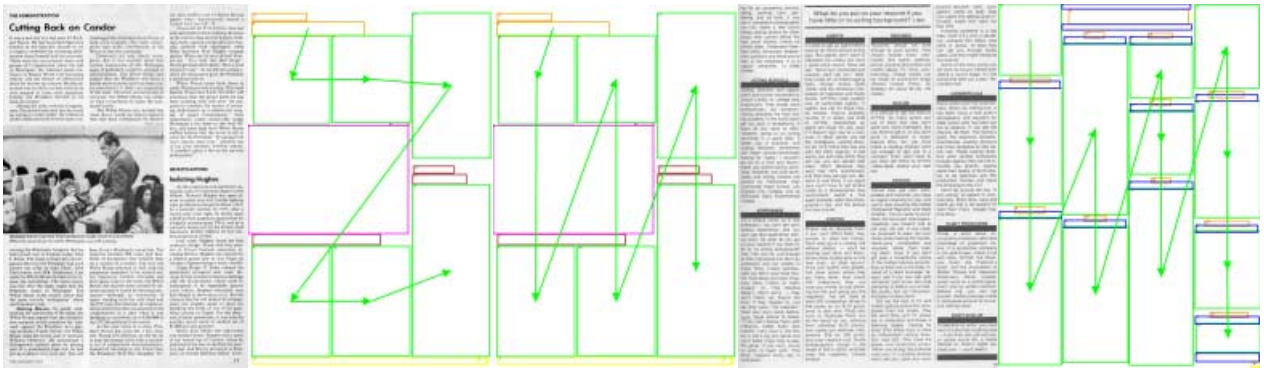
$$prec = \frac{|\mathrm{D} \cap \mathrm{CRO}|}{|\mathrm{D}|} \qquad rec = \frac{|\mathrm{D} \cap \mathrm{CRO}|}{|\mathrm{CRO}|}$$

The precision value lies between 0 and 1 inclusive, where 0 indicates that the correct reading is not among the reading orders detected, 1 indicates that there is exactly one reading order and it is correct, and any other value



a **b** **c** **d** **e**

Fig. 8. Sample images from the UW-II **a, b, c** and MTDB **d, e** data sets



a **b** **c** **d** **e**

Fig. 9a–e. Examples of reading orders detected. **a** The original document image. **b** and **c** the reading orders detected by the SpaRe module. The NLP module indicate that the correct reading order is the one presented in **b**. For the document in **d** the correct reading order **e** was already detected by the SpaRe component with no use of the NLP component

indicates the degree of uncertainty of the RoD module. Because there is only one reading order, the recall can only be 1 if the correct reading is among the ones detected, and 0 if it is not.

5.2 Logical object classification

For classification of logical objects, the statistical pattern recognition package MLC++ [17] developed at the Stanford University and SGI is used. This contains implementation of various decision tree classifiers. We used the C4.5 decision tree, which can handle heterogeneous feature vectors of enumerated and continuous values.

During experimentation, the Laplace correction as pruning technique was used. In [9] it is shown that the Laplace correction gives the best performance. The Laplace correction method biases the probability towards a uniform distribution. Specifically, in a k-class problem, if a node has m instances, c of which are from a given class, the probability assigned to the class is $\frac{c+1}{m+k}$.

We used the default threshold values of MLC++ as pruning parameters. These are 0.6925 for the lowest pruning probability, and 2 for the minimum number of instances in a leaf. One can make the system more generic by increasing the value of these pruning parameters. If we require, for instance, at least ten instances in a leaf, then many branches from the decision tree, treating specific cases, will be removed.

On the University of Washington UW-II, there are 5,364 relevant text blocks. Of those, 4,151 document objects are Body Text, 530 Caption, 619 Page Number, and 64 Title. We have randomly split the the UW-II dataset into two parts: one half as training set and the other half as test set.

In MTDB, 894 different document objects are present. Of those, 520 document objects are Body Text, 135 Caption, 166 Page Number, and 73 Title. For selecting the training and test sets we used the *leave-one-out* with rotation method proposed in [14]. This is the most appropriate method given the small size of this data set.

The classification results for the UW-II and MTDB datasets are shown in Table 2. In Table 3, the confusion matrix of the classification process for UW-II dataset is shown. Each row represents the number of blocks in the ground truth of a given type; while each column represents the type as detected by LoC. The elements outside the diagonal, 173 out of 5,364, represent classification errors.

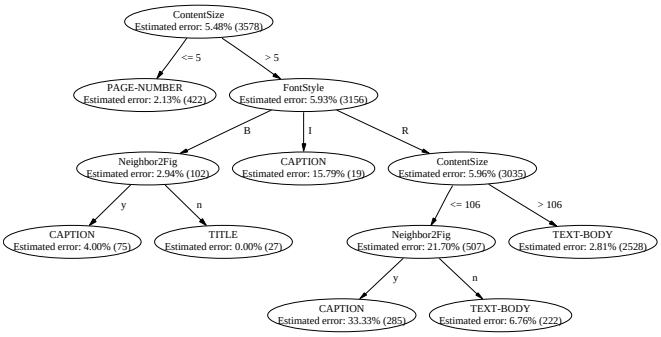


Fig. 10. The decision tree automatically generated by the MLC++. Note that this tree is heavily pruned for figure size reason. In each node, the feature name or the logical label for leaves are displayed. In parenthesis the actual number of document objects, from the test data set, that enters the tree node is displayed. The second line shows the estimated errors for the test set

Table 2. Experimental results of the LoC module

Document	*UW-II*		*MTDB*	
group	prec	rec	prec	rec
Body	0.98	0.98	0.97	0.93
Caption	0.97	0.87	0.85	0.82
PagNr	0.98	1.00	0.96	0.97
Title	0.97	0.91	0.70	0.94
total	0.98	0.94	0.87	0.92

Table 3. The confusion matrix of the classification process

	Body	*Caption*	*PageNo*	*Title*
Body	4054	85	10	2
Caption	70	460	0	0
PageNo	0	0	619	0
Title	2	4	0	58

The total wall-clock execution time[1] for the entire UW-II dataset was 93 s. For file parsing and feature vector computation 92 s were needed. The decision tree training and classification took an additional 3 s. For the MTDB dataset, file parsing and feature computation took 35 s, without OCR processing. Training and classification using decision tree is done in about 1 s.

A pruned version of the decision tree can be seen in Fig. 10. We set the minimum number of objects in a leaf at 6, compared with default value 2, and the pruning factor to 2.8 compared with default value 0.6925. Using default pruning values, we obtained results depicted in table 2. With this pruning values we not only obtain a smaller tree to fit on a page, but we also make the classification more generic. The overall error increases from 4.5% to 4.9%.

[1] The experiments were run on a PC with an Intel Pentium IV 1.7 GHz processor.

Table 4. Experimental results of the SpaRe module

Document	*UW-II*		*MTDB*	
group	prec	rec	prec	rec
trivial	0.96	1.00	0.97	0.99
regular	0.88	0.98	0.79	0.97
complex	0.93	0.98	0.88	1.00
total	0.91	0.99	0.89	0.98

5.3 Evaluation of SpaRe

In the presented experimentation, the thickness T of the boundary of the TBRR relations within the SpaRe module is set to the value of 15. The value of 15 has been shown to be best for the MTDB data set [2]. The results of experimentation with the two datasets are summarized in Table 4. On the UW-II dataset, the SpaRe module detected 672 reading orders for the 624 document pages. For 48 documents two reading orders were detected, for all other documents only one. For four complex documents the reading order detected was wrong, for five regular ones it was wrong, and for all simple documents the correct reading order was among those detected.

As for the MTDB dataset, the SpaRe module detected 192 reading orders for the 171 document pages in the data set. For 18 documents 2 reading orders were detected, instead of one. In one case, none of the two reading orders detected was correct. For one document 4 possible reading orders were detected and none of them was correct. For the remaining 152 documents, the SpaRe module detected correctly the one reading order.

The average wall-clock execution time[2] per document is of 0.11 s for the UW-II collection, while it is of 0.18 s for the MTDB collection. The overall execution time is of 66,s and 27 s for the UW-II and MTDB collections. In Fig. 11, we report the execution time in seconds of SpaRe on the two datasets with respect to the document complexity, i.e., the number of textual document objects on the page. The shape of a non-linearly growing curve is identifiable in the graphics. The small variations, such as the peak of 0.6 s at 7 for UW-II, are due to the measurement of wall-clock time. Unfortunately, the Eclipse environment does not provide primitives to measure the cpu time, but only to access the system clock. Rerunning the experiments at different moments changes these small variations preserving the overall shape of the curve.

5.4 Evaluation of the NLP module

The NLP component is only applied to pages for which SpaRe detected more than one reading order. In the current experimental setting this was the case for 50 pages from the UW-II dataset and 16 pages from the MTDB

[2] The experiments were conducted on a Sun Sparc Ultra 5 workstation with a 300 Mhz processor and other users connected.

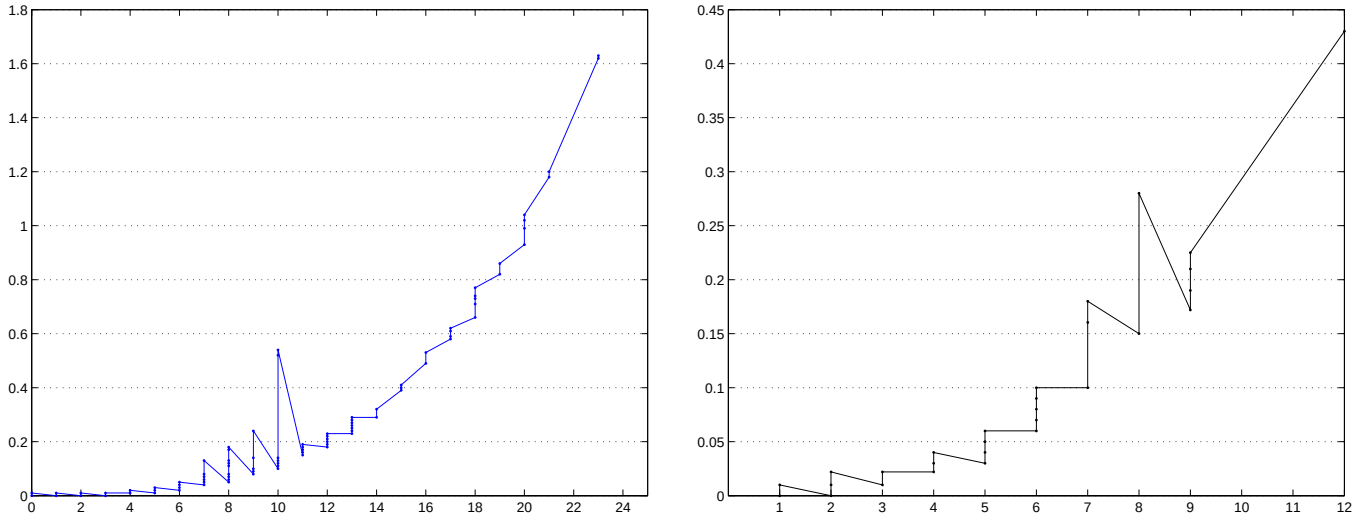


Fig. 11. The execution time per document of SpaRe on the UW-II dataset (*left*) and on the MTDB dataset (*right*) with respect to the number of textual document objects. Execution times are expressed in seconds

Table 5. Experimental results of the NLP module

	UW-II	*MTDB*
	prec	prec
total	0.82 (0.84)	0.69 (0.73)

Table 6. Experimental results of the entire system

	UW-II		*MTDB*	
	prec	rec	prec	rec
LoC	0.98	0.94	0.87	0.92
SpaRe	0.90	0.98	0.84	0.94
NLP	0.97	0.97	0.91	0.91
Total	0.97		0.91	

dataset. The experimental results for the NLP component are summarized in Table 5. Since the NLP component always returns exactly one reading order, precision and recall are always equal and therefore only the precision values are reported here.

With respect to UW-II, the NLP component yields a precision of 0.82 for all 50 pages returned by SpaRe. The precision of the NLP component when evaluated only with respect to those pages, for which SpaRe returned at least one correct reading order is 0.84. Focusing on this subset of pages more clearly indicates the actual performance of the component as it neglects cases for which it simply could not select a correct reading order.

When evaluated on MTDB, precision is 0.69 and 0.72 on the subset of pages having been assigned at least one correct potential reading order by SpaRe, as explained above.

The average execution time[3] is 2.96 s per page for UW-II, and 2.68 s per page for MTDB. This slight difference in execution time is mainly due to the higher number of text blocks per page (on average) for UW-II documents.

[3] The experiments were run on PC with Intel Pentium 800 MHz processor.

5.5 Evaluation of the entire system

The components have been evaluated in cascade for the goal of reading order detection by providing the results of the LoC module to SpaRe, and the results of the latter to the natural language module, see Fig. 7. The same evaluation measures as for assessing the individual components are used. The evaluation of the entire system is presented in Table 6. The first row shows the performance of the LoC module. Errors in this module propagate to the two reading order detection modules. The second row of the table represents the performance of the SpaRe module on the LoC input. The third row of the table represents the performance of the NLP module on the output of SpaRe. Note that since there is only one output per document of the NLP module the precision and recall measure have the same value. The final row of the module repeats the total performance of the system having as goal the reading order detection.

Comparing the second row of Table 6 with the results of SpaRe on the ground truth, cf. Table 4, one notices that the performance of the SpaRe module slightly degrades, as some document objects are misclassified. This degradation of 0.01 and 0.04 for UW-II and MTDB, respectively, is due to the fact that SpaRe attempts to de-

tect an order with elements which are then not considered to be part of a reading order. Of course, this degradation propagates to the NLP module which takes the output of the SpaRe component as its input. All of the degradation is therefore due to the fact that the reading order detected and the one found in the ground truth are not formed by the same set of document objects, not by the order of these.

The third row of Table 6 shows the results of the NLP module considering both the instances for which SpaRe returned one unique reading order and those for which the NLP selected one among those provided by SpaRe. By comparing this row with the second one, one notices that precision increases while recall decreases. The increase in precision is due to disambiguating the output of the SpaRe component for some pages, thus eliminating wrong reading orders. Dually, the overall recall slightly d ecreases, as a few correct reading orders are also eliminated by the NLP module.

5.6 Discussion

Given the goal of the presented system to process heterogeneous documents, the results presented are remarkably positive. Errors made are due to the presence of document classes not obeying the generic knowledge assumption. One could tune the system to get less errors for the given data sets, but then the generalization power is lost, and the system becomes too specific.

For logical labeling, (see Table 2), the results are better for UW-II than for MTDB. The main reason lies in the fact that in UW-II the decision tree adopted for logical labeling was trained on a larger number of document blocks. To compensate for that, we used different selection techniques for the training set. On the other hand, in MTDB the variety of documents in larger than in UW-II. These documents have different styles, and our generic decision tree can classify only the common part of them, not very specific variations.

Most of the errors are confusions among *Body*, *Caption* and *Title* objects. There are three main reasons for these errors.

First, the generic knowledge assumptions are not met for some documents. For instance, there are documents where the *Caption* is *not* a neighbor to a *Picture*. Between *Caption* and *Picture* rulers or lines were placed. See Fig. 12(a).

Second, the variety of document classes, makes the feature description of document objects less discriminative. For instance, the *Body* document objects with two or three text lines only, are confused with *Title* or *Caption* document objects. This happens mostly on UW-II dataset, where granularity of the blocks defined in ground-truth is very low.

Third, we have few errors because of wrongly detected feature vectors. In MTDB dataset some misclassifications are due to inaccuracies of the OCR, specifically, to the font characteristic detection. The font features Font-Size-Ratio and Font-Style are affected most. The errors made in character confusion by the OCR do not affect the LoC module.

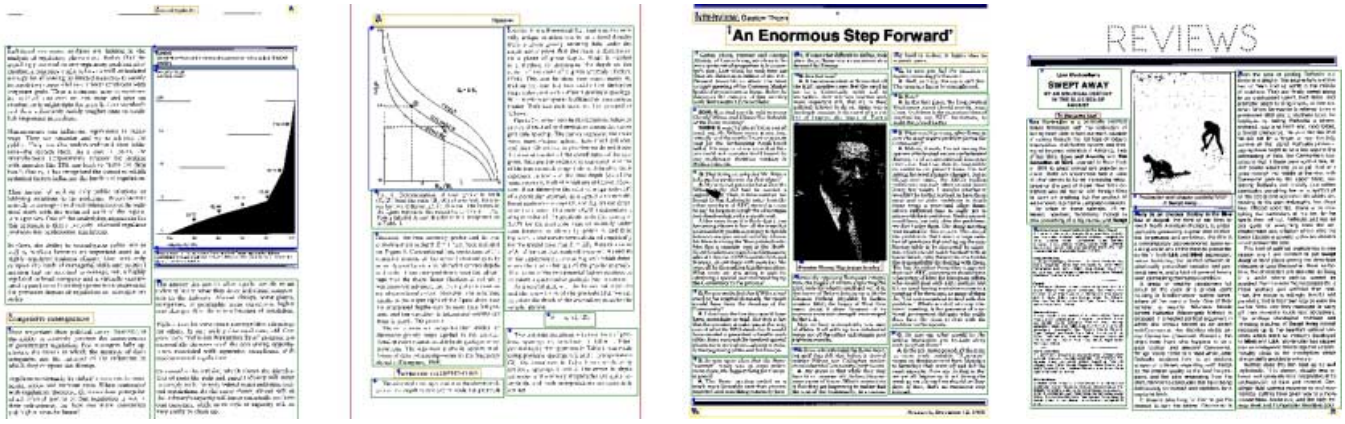
In Fig. 12 some representative pages are presented, where the logical objects classification fails, for the abovementioned reasons.

The errors made by the spatial reasoning module on the ground truth are due to the rule connecting the title to the body-reading orders. This rule is appropriate for *Main Title* but not for *(Sub)Section Title*. For instance, for a two column scientific article composed of six textual document objects, SpaRe detected four reading orders. These were all wrong because a short subtitle ("Acknowledgments") was too close to a white space in the neighboring column and was considered the title of the neighboring row in a row-wise reading. This row-wise connection was possible in four different ways, all incorrect. In case of a first page of an article in a magazine composed of three textual document objects, the title was on the left of the main text and centered vertically. In a reading order, the title was considered by SpaRe to be a subtitle of one of the two main bodies of text. It was placed incorrectly in the center of the reading order instead of on top of it. For one document composed of four textual document objects organized in one column with two subtitles and poorly typeset, SpaRe wrongly detected the reading order. The reason is that the subtitles were almost embedded in the main text and in *overlap* relation in the x axes instead of meet. The problem disappears if the threshold value is greater than 25 points.

The above suggests that better performance can be achieved by fixing the threshold for each page individually, rather than for a whole collection of documents based on some feature of the page (e.g., the white inter-column space). This should be combined with the introduction of distinct rules for main titles and section titles (cf., the case of the "Acknowledgments" subtitle).

As for the additional errors of the SpaRe module when applied to the data from the LoC module, rather than the ground truth, little can be modified. SpaRe is going to attempt to put in a spatially admissible reading order the document objects provided by the LoC module and this is never going to correspond to an actual reading order of the page.

The NLP module is invoked to resolve ambiguous situations where two or more reading orders were detected by the SpaRe module. Excluding the pages where none of the possible reading orders detected by SpaRe were correct, the average precision of the NLP module is 0.84. First, considering the UW-II dataset, six out of the eight cases, where the NLP component failed are due to 'full stop' errors. This situation arises if a text block ends with a full stop and two or more potential successors start with a letter in upper-case. In these situations n-gram models fail as they do not make any predictions about sequences of words crossing sentence boundaries. The remaining two errors are basically of the same nature, but instead of being caused by a full stop they were caused by a colon.



a **b** **c** **d**

Fig. 12a–d. Examples of pages for which the classification of logical objects fails. **a** The Caption is not neighbor to the image. **b** The Caption looks very much like a Body-Text. **c** The Body-Text on the top-left corner looks like a Title. **d** For the top-right Title the Font-Size-Ratio is 1, and all the other features, are similar to a Body Text

With respect to the MTDB dataset, three of the five errors caused by the NLP module are 'full stop' errors. For the remaining two errors, a faulty block order had been assigned a higher probability than the actual reading order. In both cases, there was no clear indication what caused these errors. It should be noted that using n-grams to compute the transitional probabilities between text blocks is a very shallow approach, having the advantage of being very efficient, but also being somewhat error prone.

On one of the pages the pages that was assigned a wrong reading order, a text block stared with an OCR error: 'atios' instead of 'ratios'. Since the n-gram probabilities are not computed on the actual words, but on their part-of-speech tags and 'atios' had been assigned the `NNS` tag (plural noun), just as it would have been the case for 'ratios', the OCR error did not affect the computation of the transitional probabilities.

The average execution time on a document image for the cascade of the three components of the system is in terms of less than 1 s using standard computers. This is a guarantee of the usability of a system based on the computer vision, artificial intelligence, and natural language processing techniques described here. However, we remark on the non-linear growth of execution time with the increase in document complexity. This may be a cause of concern when applying the system to complex documents such as big-format newspaper pages. Time-efficient solutions for very complex documents may come from the consideration of hierarchical approach (considering different granularities for document objects) and from exploiting locality (analyzing connected subsets of the document image). In addition, faster inference mechanisms can be devised for spatial reasoning. On the one hand, constraint propagation algorithms specialized for the case of bidimensional Allen relations could be devised exploiting the results in [8]. On the other hand, model checking may provide a faster alternative to constraint satisfaction techniques [1].

6 Conclusion

We have proposed a document understanding system that uses generic document knowledge for detecting the logical structure of a broad class of documents. Given the layout structure of a document, geometric information, textual features, and content are used to classify the logical objects and to detect the reading order.

The main contribution of this approach is its generality. Virtually nothing is assumed about the input document. The knowledge used is generic and applies to broad classes of documents. The current implementation is able to process English documents only, but it can be easily adapted to other languages. To process documents written in other languages, one has only to use proper part-of-speech taggers and statistical data.

The generic knowledge used in the spatial reasoning (SpaRe) component refers to Western culture, where reading order is from left-to-right and top-to-bottom. To adapt our system to styles, one has to rewrite the BeforeInReading rules, not to redesign the entire system.

The methods presented are a major step towards a full document understanding system in which all information sources are employed and little is assumed regarding the specificities of the documents.

Acknowledgements. The authors thank Arnold Smeulders for fruitful discussion and the anonymous reviewers for constructive comments. Marco Aiello is supported in part by the Italian National Research Council (CNR), grant 203.15.10, and by the Univ. of Amsterdam. Christof Monz is supported by the Physical Sciences Council with financial support from the Netherlands Organization for Scientific Research (NWO), project 612-13-001. Leon Todoran is supported by Senter den Haag and Océ Technologies BV, Venlo (IOP project IBV 96008).

References

1. M. Aiello: Document image analysis via model checking. Special issue on AI techniques in pattern recognition. AI*IA Notizie XV(1):45–48, 2002
2. M. Aiello: Spatial reasoning: theory and practice. PhD thesis, ILLC, University of Amsterdam, 2002
3. M. Aiello, C. Monz, L. Todoran: Combining linguistic and spatial information for document analysis. In: J. Mariani, D. Harman, (eds.), Proc. RIAO'2000 Content-Based Multimedia Information Access, CID, 2000, pp 266–275
4. J. Allen: Maintaining knowledge about temporal intervals. Comm ACM 26:832–843, 1983
5. O. Altamura, F. Esposito, D. Malerba: Transforming paper documents into XML format with WISDOM++. Int J Doc Anal Recognition 4(1):2–17, 2001
6. F. Aurenhammer: Voronoi diagrams – a survey of a fundamental geometric data structure. ACM Comput Surv 23(3):345–405, 1991
7. R. Baeza-Yates, B. Ribeiro-Neto: Modern information retrieval. Addison-Wesley, Reading, Mass., USA, 1999
8. P. Balbiani, J. Condotta, L. Fariñas del Cerro: A model for reasoning about bidimensional temporal relations. In: A. Cohn, L. Schubert, S. Shapiro, (eds.), Proc. 6th International Conference on Principles of Knowledge Representation and Reasoning (KR'98), pp 124–130. Morgan Kaufmann, San Francisco, 1998
9. J.P. Bradford, C. Kuntz, R. Kohavi, C. Brunk, C.E. Brodley: Pruning decision trees with misclassification costs. Technical report, Purdue University, Purdue, 1998
10. F. Cesarini, E. Francesconi, M. Gori, G. Soda: A two-level knowledge approach for understanding documents of a multi-class domain. In: ICDAR'99 pp 135–138, Bangalore, India, 1999
11. F. Cesarini, M. Gori, S. Marinai, G. Soda: Informys: a flexible invoice-like form-reader system. IEEE Trans PAMI 20(7):730–746, 1998
12. A. Dengel, F. Dubiel: Logical labeling of document images based on form layout features. In: Proc. 1997 Workshop on Document Image Analysis, DIA'97, pp 26–31, Ulm, Germany, 1997
13. Eclipse Constraint Logic Programming System: Available at: http://www-icparc.doc.ic.ac.uk/eclipse
14. A.K. Jain, P.W. Duin, J. Mao: Statistical pattern recognition: a review. IEEE Trans PAMI 22(1):4–37, 2000
15. S. Klink, A. Dengel, T. Kieninger: Document structure analysis based on layout and textual features. In: Proc. 4th IAPR International Workshop on Document Analysis Systems, DAS2000, pp 99–111, Rio de Janeiro, Brazil, 2000
16. D.E. Knuth: The art of computer programming. Addison-Wesley, Reading, Mass., USA 1973
17. R. Kohavi, D. Sommerfield, J. Dougherty: Data mining using MLC++: a machine learning library in C++. In: Proc. 8th International Conference on Tools with Artificial Intelligence. pp 234–245, 1996. Available at: http://www.sgi.com/Technology/mlc
18. D.X. Le, G.R. Thoma, H. Wechsler: Classification of binary document iamges into textual or nontextual data blocks using neural networks models. Mach Vision Appl 8:289–304, 1995
19. K.H. Lee, Y.C. Choy, S.B. Cho: Geometric structure analysis of document images: a knowledge approach. IEEE Trans PAMI 22(11):1224–1240, 2000
20. X. Li, P.A. Ng: A document classification an extraction system with learning ability. In: ICDAR'99, pp 197–200, Bangalore, India, 1999
21. C. Manning, H. Schütze: Foundations of statistical natural language processing. MIT, Boston, Mass., USA, 1999
22. A. Mukerjee, G. Joe: A qualitative model for space. In: Proc. AAAI-90, pp 721–727. AAAI, New York, 1990
23. G. Nagy: Twenty years of document image analysis in PAMI. IEEE Trans Pattern Anal Mach Intell 22(1):38–62, 2000
24. G. Nagy, J. Kanai, M. Krishnamoorthy: Two complementary techniques for digitized document analysis. In: Proc. ACM Conf. Document Processing Systems, pp 169–176, Santa Fe, N.M., USA, 1988
25. G. Nagy, S. Seth: Hierarchical representation of optically scanned documents. In: Proc. ICPR pp 350–369, 1984
26. D. Niyogi, S.N. Srihari: Using domain knowledge to derive logical structure of documents. In: Proc. SPIE, Document Recognition and Retrieval III, 1996. pp 114–125, San Jose, Calif., USA, 1996
27. G.I. Sainz Palmero, Y.A. Dimitriadis: Structured document labeling and rule extraction using a new recurent fuzzy-neural system. In: ICDAR'99 pp 181–184, Bangalore, India, 1999
28. I. Phillips, R. Haralick: Uw-ii English/Japanese document image database. CD-Rom
29. J.R. Quinlan: C4.5: Programs for machine learning. Morgan Kaufmann, San Francisco, 1993
30. J. Sauvola, H. Kauniskangas: MediaTeam Document Database II. CD-ROM collection of document images, University of Oulu, Finland. Available at: http://www.mediateam.oulu.fi/MTDB/index.html
31. F.Y. Shih, S.S. Chen: Adaptive Document Bock Segmentation and Classification. IEEE Trans Syst Man Cybern B: Cybernetics 26:797–803, 1996
32. C.K. Shin, D.S. Doermann: Classification of document page images based on visual similarity on layout structures. In: Proc- SPIE vol. 3967. In: D.P. Lopresti, J. Zhou, (eds.), Document Recognition and Retrieval VII, pp 182–190, San Jose, Calif., USA, 2000
33. R. Singh, A. Lahoti, A. Mukerjee: Interval-algebra based block layout analysis and document template generation. In: Workshop on Document Layout Interpretation and its Applications, 1999
34. TextBridge SDK 4.5: Available at: http://www.scansoft.com
35. L. Todoran, M. Aiello, C. Monz, M. Worring: Logical structure detection for hetrogeneous document classes. In: Proc. SPIE vol. 3407. In: P.B. Kantor, D.P. Lopresti, J. Zhou, (eds.), Document Recognition and Retrieval VIII, pp 99–111, San Jose, Calif., USA, 2001
36. L. Todoran, M. Worring, A.M.W Smeulders: The UvA color document dataset. Technical Report 2002-01, UvA, Amsterdam, 2002 IJDAR (submitted)
37. S. Tsujimoto, H. Asada: Major components of a complete text reading system. Proc. IEEE 80(7):1133–1149, 1992
38. H. Walischewski: Automatic knowledge acquisition for spatial document interpretation. In: Proc. 4th ICDAR'97 pp 243–247, Ulm, Germany, 1997
39. D. Wang, S.N. Srihari: Classification of newspaper image blocks using texture analysis. Comput Vision Graphics Image Process 47:327–352, 1989

40. Y. Wang, R. Haralick, I.T. Phillips: Zone content classification and its performance evaluation. In: Proc. ICDAR 2001 pp 540–544, Seattle, Wash., USA, 2001

Marco Aiello received his M.Sc. in engineering and computer science from the University of Rome La Sapienza (cum laude) in 1997, and his PhD from the University of Amsterdam on spatial reasoning formalisms and their application in 2002. Currently he is a contract researcher with the Department of Information and Communication Technologies at the University of Trento, Italy. His research interests include qualitative spatial reasoning, document understanding, image retrieval, and distributed systems.

Christof Monz received a degree in computational linguistics from the University of Stuttgart, Germany, in 1999. He is currently a PhD student in computer science at the University of Amsterdam. His research interests include information retrieval, corpus-based question answering, and statistical natural language processing.

Leon Todoran received his M.Sc. degree in computer science from Technical University, Cluj-Napoca, Romania in 1993. Between 1993 and 1997 he worked as teaching assistant at Technical University Cluj-Napoca. He is currently a PhD Student at the University of Amsterdam, The Netherlands. His research interests include image processing, document analysis, and pattern recognition. He is a student member of the IEEE, Computer Society.

Marcel Worring received a degree in computer science (honors) from the Free University Amsterdam. His Ph.D. was on digital image analysis and was obtained from the University of Amsterdam, The Netherlands in 1993. He became an Assistant Professor in 1995 in the Intelligent Sensory Information Systems group at the University of Amsterdam. His current interests are in multimedia information analysis in particular document and video analysis. He has been a visiting researcher in the Department of Diagnostic Imaging at Yale University (1992) and at the Visual Computing Lab at the University of California, San Diego (1998). He was a co-organizer of the Document Layout Interpretation and its Applications workshops in Bangalore ('99) and Seattle ('01).