Machine Learning for Reading Order Detection in Document Image Understanding

Donato Malerba, Michelangelo Ceci, and Margherita Berardi

Dipartimento di Informatica, Università degli Studi di Bari
via Orabona, 4 - 70126 Bari - Italy
{malerba,ceci,berardi}@di.uniba.it

Summary. Document image understanding refers to logical and semantic analysis of document images in order to extract information understandable to humans and codify it into machine-readable form. Most of the studies on document image understanding have targeted the specific problem of associating layout components with logical labels, while less attention has been paid to the problem of extracting relationships between logical components, such as cross-references. In this chapter, we investigate the problem of detecting the reading order relationship between components of a logical structure. The domain specific knowledge required for this task is automatically acquired from a set of training examples by applying a machine learning method. The input of the learning method is the description of "chains" of layout components defined by the user. The output is a logical theory which defines two predicates, $first\_to\_read/1$ and $succ\_in\_reading/2$, useful for consistently reconstructing all chains in the training set. Only spatial information on the page layout is exploited for both single and multiple chain reconstruction. The proposed approach has been evaluated on a set of document images processed by the system WISDOM++.

1 Introduction

Documents are characterized by two important structures: the layout structure and the logical structure. Both are the results of repeatedly dividing the content of a document into increasingly smaller parts, and are typically represented by means of a tree structure. The difference between them is the criteria adopted for structuring the document content: the layout structure is based on the *presentation* of the content, while the logical structure is based on the *human-perceptible meaning* of the content.

The extraction of the layout structures from images of scanned paper documents is a complex process, typically denoted as *document layout analysis*, which involves several steps including preprocessing, page decomposition (or segmentation), classification of segments according to content type (e.g., text, graphics, pictures) and hierarchical organization on the basis of perceptual

D. Malerba et al.: *Machine Learning for Reading Order Detection in Document Image Understanding*, Studies in Computational Intelligence (SCI) **90**, 45–69 (2008)
www.springerlink.com

criteria. *Document image understanding* refers to the process of extracting the logical structure of a document image. This task is strongly application dependent, since the same definition of the logical structure depends on the type of information the user is interested in retrieving in a document.

Most works on document image understanding aim at associating a "logical label" with some components of the layout structure: this corresponds to mapping (part of) the layout structure into the logical structure. Generally, this mapping is based on spatial properties of the layout components, such as absolute positioning with respect to a system of coordinates, relative positioning (e.g., on top, to right), geometrical properties (e.g., height and width), as well as information on the content type (e.g., text, graphics, and picture). Some studies have also advocated the use of textual information of some layout components to base, or at least to refine, the classification of layout components into a set of logical labels.

The main problem for all these approaches remains the large amount of domain specific knowledge required to effectively perform this task. Hand-coding the necessary knowledge according to some formalism, such as block grammars [1], geometric trees [2], and frames [3] is time-consuming and limits the application of a document image understanding system to a set of predefined classes of documents. To alleviate the burden in developing and customizing document image understanding systems, several data mining and machine learning approaches have been proposed with the aim of automatically extracting the required knowledge [4].

In its broader sense, document image understanding cannot be considered synonymous of "logical labeling", since relationships among logical components are also possible and their extraction can be equally important for an application domain. Some examples of relations are the cross reference of a caption to a figure, as well as the cross reference of an affiliation to an author. An important class of relations investigated in this chapter is represented by the *reading order* of some parts of the document. More specifically, we are interested in determining the reading order of most abstract layout components on each page of a multi-page document. Indeed, the spatial order in which the information appears in a paper document may have more to do with optimizing the print process than with reflecting the logical order of the information contained.

Determining the correct reading order can be a crucial problem for several applications. By following the reading order recognized in a document image, it is possible to cluster together text regions labelled with the same logical label into the same textual component (e.g., "introduction", "results", "method" of a scientific paper). Once a single textual component is reconstructed, advanced techniques for text processing can be subsequently applied. For instance, information extraction methods may be applied locally to reconstructed textual components of documents (e.g., sample of the experimental setting studied in the "results" section). Moreover, retrieval of document images on the basis of their textual contents is more effectively realized.

Several papers on reading order detection have already been published in the literature. Their brief description is provided in the next Section. Some are based only on the spatial properties of the layout components, while others also exploit the textual content of parts of documents. Moreover, some methods have been devised for properly ordering layout components (independent of their logical meaning), while others consider the recognition of some logical components, such as "title" and "body", as preliminary to reading order detection. A common aspect of all methods is that they strongly depend on the specific domain and are not "reusable" when the classes of documents or the task at hand change.

As for logical labelling, domain specific knowledge required to effectively determine the reading order can be automatically acquired by means of machine learning methods. In this study we investigate the problem of inducing rules which are used for predicting the proper reading order of layout components detected in document images. The rules are learned from training examples which are sets of ordered layout components described by means of both their spatial properties and their possible logical label. Therefore, no textual information is exploited to understand document images. The ordering of the layout components is defined by the user and does not necessarily reflect the traditional Western-style document encoding rule according to which reading proceeds top-bottom and left-right. For instance, the user can specify a reading order according to which the affiliation of an author immediately follows the author's name, although the two logical components are spatially positioned quite far away on the page layout (e.g., the affiliation is reported at the bottom of the first column of the paper). In multi-page articles, such as those considered in this chapter, ordering is defined at the page level. More precisely, different "chains" of layout components can be defined by the user, when independent pieces of information are represented on the same page (e.g., the end of an article and the beginning of a new one). Chains are mutually exclusive, but not necessarily exhaustive, sets of most abstract layout components in a page, so that their union defines a partial (and not necessarily a total) order on the set of layout objects.

This chapter is organized as follows. In the next section, the background and some related works are reported, while the reading order problem is formally defined in Section 3. The machine learning system applied to the problem of learning from ordered layout components is introduced in Section 4. The representation of training examples as well as the manner in which learned rules are applied to a new document are also illustrated. Some experimental results on a set of multi-page printed documents are reported and commented on in Section 5. Finally, Section 6 concludes and discusses ideas for further studies.

2 Background and Related Works

In the literature there are already several publications on reading order detection. A pioneer work is reported in [5], where multi-column and multi-article documents (e.g., magazine pages) with figures and photographs are handled. Each document page is described as a tree, where each node, except the root, represents a set of adjacent blocks located in the same column, ordered so that the block on the upper location precedes the others. Direct descendants of an internal node are also ordered in sequence according to their locations in the same way that the block to the left and on the top precedes the others. Reading order detection follows a preliminary rough classification of layout components into "title" and "body". Heads are blocks in which there are only a few text lines with large type fonts, while bodies correspond to blocks with several text lines with small type fonts. The reading order is extracted by applying some hand-coded rules which allow the transformation of trees representing layout structures (with associated 'title" and "body" labels) into ordered structures. Once the correct reading order is detected, a further interpretation step is performed to attach some logical labels (e.g., title, abstract, sub-title, paragraph) to each item of the ordered structure.

A similar tree-structured representation of the page layout is adopted in the work by Ishitani [6]. The structure is derived by a recursive XY-cut approach [7], that is, a recursive horizontal/vertical partitioning of the input image. The XY-cut process naturally determines the reading order of the layout components, since for horizontal cuts the top-bottom ordering is applied to the derived sections, while for vertical cuts the right-left (i.e., Japanese style) ordering is applied to the derived columns.

The main problem with this XY-cut approach is that at each recursion step, there are often multiple possible, and possibly conflicting, cuts. In the original algorithm, the widest cut is selected at each recursion. While this strategy works reasonably well for a page segmentation task, it is not always appropriate for a reading order detection task. For this reason, Ishitani proposed a bottom-up approach using three heuristics which take into account local geometric features, text orientation and distance among vertically adjacent layout objects in order to merge some layout objects before performing the XY-cut. As observed by Meunier [8], this aims at reducing the probability of having to face multiple cutting alternatives, but it does not truly prevent them from occurring. For this reason, he proposed to reformulate the problem of recursively cutting a page as an optimization problem, and defined both a scoring function for alternative cuts, and a computationally tractable method for choosing the best partitioning.

A common aspect of all these approaches is that they are based exclusively on the spatial information conveyed by a page layout. On the contrary, Taylor et al. [9], propose the use of linguistic information to define the proper reading order. For instance, to determine whether an article published in a magazine

continues on the next page, it is suggested to look for a text, such as 'continued on next page'.

The usage of linguistic information has also been proposed by Aiello et al. [10], who described a document analysis system for logical labelling and reading order extraction of broad classes of documents. Each document object is described by means of both attributes (i.e., aspect ratio, area ratio, font size ratio, font style, content size, number of lines) and spatial relations (defined as extensions of Allen's interval relations [11]). Only objects labelled with some logical labels (title and body) are considered for reading order. More precisely, two distinct reading orders are first detected for the document object types *Title* and *Body*, and then they 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*. Each reading order is determined in two steps. Initially, spatial information on the document objects is exploited by a spatial reasoner which solves a constraint-satisfaction problem, where constraints correspond to general document encoding rules (e.g., "in the Western-culture, documents are usually read top-bottom and left-right"). The output of the spatial reasoner is a (cyclic) graph where edges represent instances of the partial ordering relation *BeforeInReading*. A reading order is then defined as a full path in this graph, and is determined by means of an extension of a standard topological sort [12]. Due to the generality of the document encoding rule used by the spatial reasoner, it is likely that one obtains more than one reading order, especially for complex documents with many blocks. For this reason, a natural language processor is used in the second step of the proposed method. The goal is that of disambiguating between different reading orders on the basis of textual information of logical objects. This step works by computing probabilities of sequences of words obtained by joining document objects which are candidates to be followed in reading. The best aspect of this work is the generality of the approach due to the generality of the knowledge adopted in reasoning.

Topological sorting is also exploited in the approach proposed by Breuel [13]. In particular, reading order is defined the basis of text lines segments, which are pairwise compared on the basis of four simple rules in order to determine a partial order. Then a topological sorting algorithm is applied to find at least one global order consistent with this partial order. Columns, paragraphs, and other layout features are determined on the basis of the spatial arrangement of text line segments in reading order. For instance, paragraph boundaries are indicated by relative indentation of consecutive text lines in reading order.

All approaches reported above reflect a clear domain specificity. For instance, the classification of blocks as "title" and "body" is appropriate for magazine articles, but not for administrative documents. Moreover, the document encoding rules appropriate for Western-style documents are different for Japanese papers. Surprisingly, there is no work, to the best of our knowledge, that handles the reading order problem by resorting to machine learning

techniques, which can generate the required knowledge from a set of training layout structures whose correct reading order has been provided by the user. In previous works on document image analysis and understanding, we investigated the application of machine learning techniques to several knowledge-based document image processing tasks, such as classification of blocks according to their content type [14], automatic global layout analysis correction [15], classification of documents into a set of pre-defined classes [16], and logical labelling [17]. Experimental results always proved the feasibility of this approach, at least on a small scale, that is, for a few hundred of training document images. Therefore, following this mainstream of research, herein we consider the problem of learning the definition of reading order.

The proposed solution has been tested by processing documents with WISDOM++[1], a knowledge-based document image processing system originally developed to transform multi-page printed documents into XML format. WISDOM++ makes extensive use of knowledge and XML technologies for semantic indexing of paper documents. This is a complex process involving several steps:

1. The image is segmented into basic layout components (basic blocks), which are classified according to the type of content (e.g., text, pictures and graphics).
2. A perceptual organization phase (layout analysis) is performed to detect a tree-like layout structure, which associates the content of a document with a hierarchy of layout components.
3. The first page is classified to identify the membership class (or type) of the multi-page document (e.g. scientific paper or magazine).
4. The layout structure of each page is mapped into the logical structure, which associates the content with a hierarchy of logical components (e.g. title or abstact of a scientific paper).
5. OCR is applied only to those logical components of interest for the application domain (e.g., title).
6. The XML file that represents the layout structure, the logical structure, and the textual content returned by the OCR for some specific logical components is generated.
7. XML documents are stored in a repository for future retrieval purposes.

Four of seven processing steps make use of explicit knowledge expressed in the form of decision trees and rules which are automatically learned by means of two distinct machine learning systems: ITI [18], which returns decision trees useful for block classification (first step), and ATRE [19], which returns rules for layout analysis correction (second step) [15], document image classification (third step) and document image understanding (fourth step) [4]. As explained in Section 4, ATRE is also used to learn the intensional definition of two

[1] http://www.di.uniba.it/∼malerba/wisdom++/

predicates, which contribute to determine the reading order chains in a page layout.

3 Problem Definition

In order to formalize the problem we intend to solve, some useful definitions are necessary:

Definition 1. *Partial Order [20]*
Let A be a set of blocks in a document page, a partial order P over A is a relation $P \in A \times A$ such that P is

1. *reflexive* $\forall s \in A \Rightarrow (s, s) \in P$
2. *antisymmetric* $\forall s_1, s_2 \in A$: $(s_1, s_2) \in P \wedge (s_2, s_1) \in P \Leftrightarrow s_1 = s_2$
3. *transitive* $\forall s_1, s_2, s_3 \in A$: $(s_1, s_2) \in P \wedge (s_2, s_3) \in P \Rightarrow (s_1, s_3) \in P$

Definition 2. *Weak Partial Order*
Let A be a set of blocks in a document page, a weak partial order P over A is a relation $P \in A \times A$ such that P is

1. *irreflexive* $\forall s \in A \Rightarrow (s, s) \notin P$
2. *antisymmetric* $\forall s_1, s_2 \in A$: $(s_1, s_2) \in P \wedge (s_2, s_1) \in P \Leftrightarrow s_1 = s_2$
3. *transitive* $\forall s_1, s_2, s_3 \in A$: $(s_1, s_2) \in P \wedge (s_2, s_3) \in P \Rightarrow (s_1, s_3) \in P$

Definition 3. *Total Order*
Let A be a set of blocks in a document page, a partial order T over the set A is a total order iff $\forall s_1, s_2 \in A$: $(s_1, s_2) \in T \vee (s_2, s_1) \in T$

Definition 4. *Complete chain*
Let:

- *A be a set of blocks in a document page,*
- *D be a weak partial order over A*
- *$B = \{a \in A | (\exists b \in A \text{ s.t. } (a, b) \in D \vee (b, a) \in D)\}$ be the subset of elements in A related to any element in A itself.*

If $D \cup \{(a, a) | a \in B\}$ is a total order over B, then D is a complete chain over A

Definition 5. *Chain reduction*
Let D be a complete chain over A
the relation
$C = \{(a, b) \in D | \neg \exists c \in A \text{ s.t. } (a, c) \in D \wedge (c, b) \in D\}$
is the reduction of the chain D over A.

Example 1. Let $A = \{a, b, c, d, e\}$. If $D = \{(a, b), (a, c), (a, d), (b, c), (b, d), (c, d)\}$ is a complete chain over A, then $C = \{(a, b), (b, c), (c, d)\}$ is its reduction (see Figure 1).

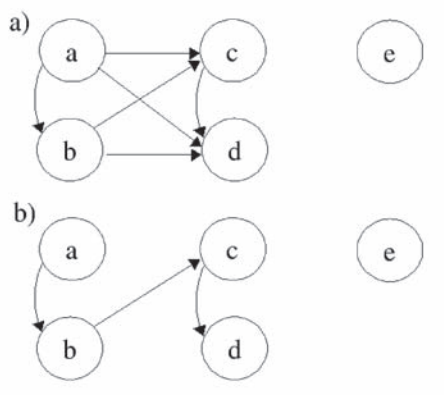

Fig. 1. A complete chain (a) and its reduction (b)

Indeed, for our purposes it is equivalent to deal with complete chains or their reduction. Henceforth, for the sake of simplicity, the term *chain* will denote the reduction of a complete chain.

By resorting to the definitions above, it is possible to formalize the reading order induction problem as follows:

Given :

- A description $DesTP_i$ in the language L of the set of n training pages $TrainingPages = \{TP_i \in \Pi | i = 1..n\}$ (where Π is the set of pages).
- A description $DesTC_i$ in the language L of the set TC_i of chains (over $TP_i \in TrainingPages$) for each $TP_i \in TrainingPages$.

Find :

An intensional definition T in the language L of a chain over a generic page $P \in \Pi$ such that T is complete and consistent with respect to all training chains descriptions $DesTC_i$, $i = 1..n$.

In this problem definition, we refer to the intensional definition T as a first order logic theory. The fact that T is complete and consistent with respect to all training chains descriptions can be formally described as follows:

Definition 6 (Completeness and Consistency).
Let:

- *T be a logic theory describing chains instances expressed in the language L,*
- *E^+ be the set of positive examples for the chains instances ($E^+ = \bigcup_{i=1..n} \bigcup_{TC \in TC_i} TC$),*
- *E^- be the set of negative examples for the chains instances ($E^- = \bigcup_{i=1..n} (TP_i \times TP_i)/E^+$),*

- $DesE^+$ *be the description of* E^+ *in* L,
- $DesE^-$ *be the description of* E^- *in* L,

then T *is complete and consistent with respect to all training chains descriptions iff* $T \models DesE^+ \ \wedge \ T \not\models DesE^-$.

This formalization of the problem permits us to represent and identify distinct reading orders on the same page and avoids including blocks that should not be included in the reading order (e.g. figures or page numbers).

4 Learning Reading Order

In order to learn first order logic theories for reading order identification, in this work we use the learning system ATRE[2] [19]. Indeed, ATRE is particularly suited for the task at hand since the spatial dimension of page layout makes methods developed in inductive logic programming (ILP) [21, 22, 23, 24, 25] the most suitable approaches. This conviction comes from the fact that most of the classical learning systems assume that training data are represented in a single table of a relational database, such that each row (or tuple) represents an independent example (a layout component) and columns correspond to properties of the example (e.g., height of the layout component). This single-table assumption, however, is quite naive for at least three reasons. First, layout components cannot be realistically considered independent observations, because their spatial arrangement is mutually constrained by formatting rules typically used in document editing. Second, spatial relationships between a layout component and a variable number of other components in its neighborhood cannot be properly represented by a fixed number of attributes in a table. Third, different layout components may have different properties (e.g., the property "brightness" is appropriate for half-tone images, but not for textual components), so that properties of the components in the neighborhood cannot be effectively represented by the same set of attributes. Since the single-table assumption limits the representation of relationships (spatial or non) between examples, it also prevents the discovery of this kind of pattern which could be very useful in reading order identification.

To automate the reconstruction of the reading order, WISDOM++ has been opportunely extended in order to:

- Allow the user to define correct reading order chains on training documents through the visual interaction with the system.
- Represent training reading order chains in first order logic formalism which the learning system is able to interpret.
- Run the learning system.
- Apply the learned theories on new (testing) document images.

[2] http://www.di.uniba.it/∼malerba/software/atre

In the following, we briefly present the learning system ATRE. Subsequently, we present the document descriptions provided by WISDOM++ to the learning system. Lastly, we explain how knowledge acquired by the learning system is exploited for order reconstruction.

4.1 ATRE: The Learning System

ATRE is an ILP system which can learn recursive theories from examples. The learning problem solved by ATRE can be formulated as follows:

Given

- a set of *concepts* $C_1, C_2, \ldots, C_r$ to be learned
- a set of *observations* O described in a language L_O
- a *background theory* BK
- a *language* of hypotheses L_H
- a user's *preference criterion* PC

Find

A logical theory T expressed in the language L_H and defining the concepts $C_1, C_2, \ldots, C_r$, such that T is complete and consistent with respect to O and satisfies the preference criterion PC.

The *completeness* property holds when the theory T explains all observations in O of the r concepts $C_1, C_2, \ldots, C_r$, while the *consistency* property holds when the theory T explains no counter-example in O of any concept C_i. The satisfaction of these properties guarantees the correctness of the induced theory with respect to the given observations O. Whether the theory T is correct with respect to additional observations not in O is an extra-logical matter, since no information on the generalization accuracy can be drawn from the training data themselves. In fact, the selection of the "best" theory is always made on the grounds of an inductive bias embedded in some heuristic function or expressed by the user of the learning system (preference criterion).

In the context of the reading order learning, we identified two concepts to be learned, namely $first\_to\_read/1$ and $succ\_in\_reading/2$. The former refers to the first layout component of a chain, while the latter refers to the relation *successor* between two components in a chain. By combining the two concepts it is possible to identify a partial ordering of blocks in a document page.

As to the representation languages, the basic component is the *literal*, which can be of the two distinct forms:

$f(t_1, \ldots, t_n) =$ Value (simple literal)

$f(t_1, \ldots, t_n) \in$ Range (set literal),

where f and g are function symbols called *descriptors*, t_i's are *terms* (constants or variables) and *Range* is a closed interval of possible values taken by f. In

the next section descriptors used for the representation of training examples and hypotheses are presented.

4.2 Document Description

In ATRE, training observations are represented by ground multiple-head clauses [26], called *objects*, which have a conjunction of simple literals in the head. The head of an object contains positive and negative examples for the concepts to be learned, while the body contains the description of layout components on the basis of geometrical features (e.g. width, height) and topological relations (e.g. vertical and horizontal alignments) existing among blocks, the type of the content (e.g. text, horizontal line, image) and the logic type of a block (e.g. title or authors of a scientific paper). Terms of literals in objects can only be constants, where different constants represent distinct layout components within a page.

The complete list of descriptors used for this task is reported in Table 1.

Table 1. Descriptors used in the first-order representation of the layout components

Descriptor name	Definition
width(block)	Integer domain
height(block)	Integer domain
x_pos_centre(block)	Integer domain
y_pos_centre(block)	Integer domain
part_of(block1,block2)	Boolean domain: true if block1 contains block2
alignment(block1,block2)	Nominal domain: only_left_col, only_right_col, only_middle_col, only_upper_row, only_lower_row, only_middle_row
to_right(block1,block2)	Boolean domain
on_top(block1,block2)	Boolean domain
type_of(block)	Nominal domain: text, hor_line, image, ver_line, graphic, mixed
<logic_type>(block)	Boolean domain: true if block is labeled as <logic_type> associated to the block
class(doc)	Nominal domain: represents the class associated to the document page
page(doc)	Nominal domain: first, intermediate, last_but_one, last. Represents the page associated to the document page.

The descriptor *<logic_type>(block)* is actually a meta-notation for a class of first order logic predicates, which depend on the application at hand. In particular, *<logic_type>* can be instantiated to the logic labels associated with layout components. In the case of scientific papers, among others, relevant

logical labels could be title, author, abstract. This means that *title*(*block*), *author*(*block*) and *abstract*(*block*) are possible descriptors.

In order to explain the semantics of geometrical and topological descriptors, some useful notation is introduced. Let us to consider the reference system whose origin is in the top left corner of the document page as shown in Figure 2. $TL_x(z)$ ($TL_y(z)$) denotes the abscissa (ordinate) of the top left corner of a (rectangular) block z, while $BR_x(z)$ ($BR_y(z)$) denotes the abscissa (ordinate) of the bottom right corner of z. Then the semantics of descriptors in Table 1 is the following:

- ***width(block)*** represents the block width, that is,
  $width(z) = BR_x(z) - TL_x(z)$
- ***height(block)*** represents the block height, that is,
  $height(z) = BR_y(z) - TL_y(z)$
- ***x_pos_centre(block)*** represents the abscissa of the block's centroid, that is, $x\_pos\_centre(z) = (BR_x(z) + TL_x(z))/2$
- ***y_pos_centre(block)*** represents the ordinate of the block's centroid, that is, $y\_pos\_centre(z) = (BR_y(z) + TL_y(z))/2$
- ***part_of(block1,block2)*** represents the fact that the layout component *block1* corresponding to the page contains the layout object *block2*, that is,
  $part\_of(z1, z2) = true \Longleftrightarrow^{def}$
  $TL_x(z1) \leq TL_x(z2) \wedge TL_y(z1) \leq TL_y(z2) \wedge BR_x(z1) \geq BR_x(z2) \wedge BR_y(z1) \geq BR_y(z2)$
- ***alignment(block1,block2)*** represents either horizontal or vertical alignment between two blocks. Six possible nominal values are considered:

  alignment(z1,z2)= only_left_col $\Longleftrightarrow^{def}$
  $abs(TL_x(z1) - TL_x(z2)) \leq \alpha \wedge (TL_y(z1) \leq TL_y(z2))$

  alignment(z1,z2)= only_right_col $\Longleftrightarrow^{def}$
  $abs(BR_x(z1) - BR_x(z2)) \leq \alpha \wedge (TL_y(z1) \leq TL_y(z2))$

  alignment(z1,z2)= only_middle_col $\Longleftrightarrow^{def}$
  $alignment(z1, z2) \neq only\_left\_col \wedge alignment(z1, z2) \neq only\_right\_col \wedge abs(x\_pos\_centre(z1) - x\_pos\_centre(z2)) \leq \alpha \wedge (TL_y(z1) \leq TL_y(z2))$

  alignment(z1,z2)= only_upper_row $\Longleftrightarrow^{def}$
  $abs(TL_y(z1) - TL_y(z2)) \leq \alpha \wedge (TL_x(z1) \leq TL_x(z2))$

  alignment(z1,z2)= only_lower_row $\Longleftrightarrow^{def}$
  $abs(BR_y(z1) - BR_y(z2)) \leq \alpha \wedge (TL_x(z1) \leq TL_x(z2))$

  alignment(z1,z2)= only_middle_row $\Longleftrightarrow^{def}$
  $alignment(z1, z2) \neq only\_upper\_row \wedge alignment(z1, z2) \neq only\_lower\_row \wedge$

$abs(y\_pos\_centre(z1) - y\_pos\_centre(z2)) \leq \alpha \wedge (TL_x(z1) \leq TL_x(z2))$

- ***on_top(block1,block2)*** indicates that $block1$ is above $block2$, that is,
  $on\_top(z1, z2) \Longleftrightarrow^{def}$
  $BR_y(z1) < TL_y(z2) \leq \beta + BR_y(z1) \wedge$
  $(TL_x(z1) \leq x\_pos\_centre(z2) \leq BR_x(z1) \vee$
  $TL_x(z2) \leq x\_pos\_centre(z1) \leq BR_x(z2))$
- ***to_right(block1,block2)*** aims at representing the fact that $block2$ is positioned to the right of $block1$. Its formal definition is:
  $to\_right(z1, z2) \Longleftrightarrow^{def}$
  $BR_x(z1) < TL_x(z2) \leq \gamma + BR_x(z1) \wedge$
  $(TL_y(z1) \leq y\_pos\_centre(z2) \leq BR_y(z1) \vee$
  $TL_y(z2) \leq y\_pos\_centre(z1) \leq BR_y(z2))$.

The last three descriptors are parametrized, that is, their semantics is based on few constants (α, β and γ) whose specification is domain dependent.

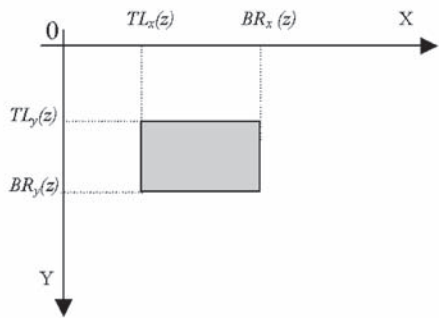

Fig. 2. Reference system used to represent components in the document page. Origin represents the origin in the top left corner of the document page

The description of the document page reported in Figure 3 is reported in the following:

$object('tpami17\_1\text{-}13', [class(p) = tpami,$
$first\_to\_read(0) = true, first\_to\_read(1) = false, ...$
$succ\_in\_reading(0, 1) = true, succ\_in\_reading(1, 2) = true,$
$..., succ\_in\_reading(7, 8) = true,$
$succ\_in\_reading(2, 10) = false, ...,$
$succ\_in\_reading(2, 5) = false],$
$[part\_of(p, 0) = true, ...,$
$height(0) = 83, height(1) = 11, ...$
$width(0) = 514, width(1) = 207, ...,$
$type\_of(0) = text, ..., type\_of(11) = hor\_line,$
$title(0) = true, author(1) = true,$
$affiliation(2) = true, ..., undefined(16) = true, ...$

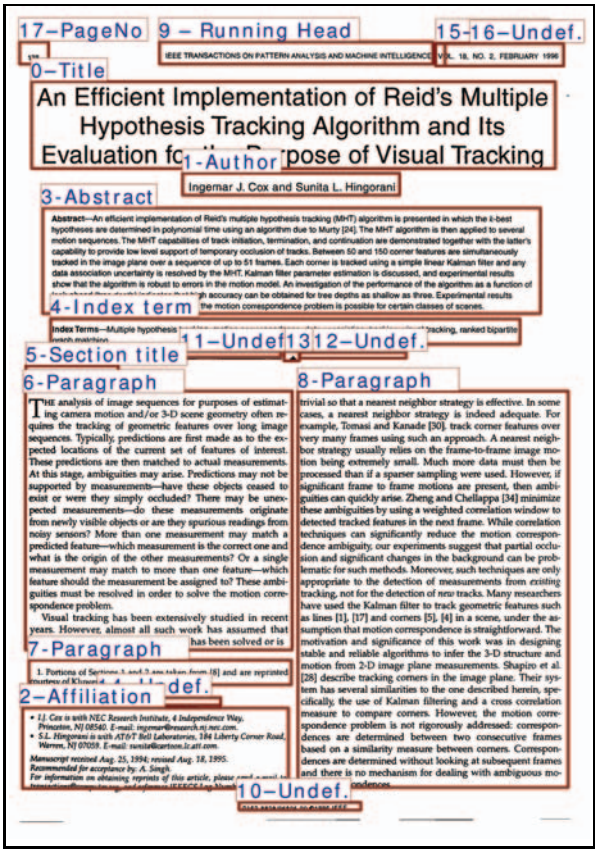

Fig. 3. A document page: For each layout component, both logical labels and constants are shown

$$x\_pos\_centre(0) = 300, x\_pos\_centre(1) = 299, ...,$$
$$y\_pos\_centre(0) = 132, y\_pos\_centre(1) = 192, ...,$$
$$on\_top(9, 0) = true, on\_top(15, 0) = true, ...,$$
$$to\_right(6, 8) = true, to\_right(7, 8) = true, ...$$
$$alignment(16, 8) = only\_right\_col, alignment(17, 5) = only\_left\_col, ...$$
$$alignment(15, 16) = only\_middle\_row,$$
$$class(p) = tpami, page(p) = first]).$$

The constant p denotes the whole page while the remaining integer constants (0, 1, ..., 17) identify distinct layout components. In this example, the block number 0 corresponds to the first block to read ($first\_to\_read(0) = true$), it is a textual component ($type\_of(0) = text$) and it is logically labelled as 'title' ($title(0) = true$). Block number 1 (immediately) follows block 0 in the reading order ($succ\_in\_reading(0, 1) = true$); it is a textual component and it includes information on the authors of the paper ($author(1) = true$).

The expressive power of ATRE is also exploited in order to define background knowledge. In this application domain, the following background knowledge has been defined:

$at\_page(X) = first \leftarrow part\_of(Y, X) = true, page(Y) = first.$
$at\_page(X) = intermediate \leftarrow part\_of(Y, X) = true, page(Y) = intermediate.$
$at\_page(X) = last\_but\_one \leftarrow part\_of(Y, X) = true, page(Y) = last\_but\_one.$
$at\_page(X) = last \leftarrow part\_of(Y, X) = true, page(Y) = last.$
$alignment(X, Y) = both\_rows \leftarrow alignment(X, Y) = only\_lower\_row,$
$\quad alignment(X, Y) = only\_upper\_row.$
$alignment(X, Y) = both\_columns \leftarrow alignment(X, Y) = only\_left\_col$
$\quad alignment(X, Y) = only\_right\_col.$

The first four rules allow information on the page order to be automatically associated to layout components, since their reading order may depend on the page order. The last two clauses define the alignment by both rows/columns of two layout components.

As explained in the previous section, ATRE learns a logical theory T defining the concepts $first\_to\_read/1$ and $succ\_in\_reading/2$, such that T is complete and consistent with respect to the examples. This means that it is necessary to represent both positive and negative examples and the representation of negative examples for the concept $succ\_in\_reading/2$ poses some feasibility problems due to their quadratic growth. In order to reduce the number of negative examples, we resort to sampling techniques. Indeed, this is a common practice in the presence of unbalanced datasets [27]. In our case, we sampled negative examples by limiting their number to 1000% of the number of positive examples. In this way, it is possible to simplify the learning stage and to have rules that are less specialized and avoid overfitting problems.

In summary, generated descriptions permit us to describe both the layout structure, the logical structure and the reading order chains of a single document page (see Figure 4).

ATRE is particularly indicated for our task since it can identify dependencies among concepts to be learned or even recursion. Examples of rules that ATRE is able to extract are reported in the following:

$first\_to\_read(X1) = true \leftarrow title(X1) = true,$
$\quad x\_pos\_centre(X1) \in [293..341], succ\_in\_reading(X1, X2) = true$
$succ\_in\_reading(X2, X1) = true \leftarrow on\_top(X2, X1) = true,$
$\quad y\_pos\_centre(X2) \in [542..783]$

The first rule states that the first block to read is a logical component labelled as title, positioned approximately at the center of the document and followed by another layout component in the reading order. The second rule states that a block $X2$ "follows in reading" another block $X1$ if it is above

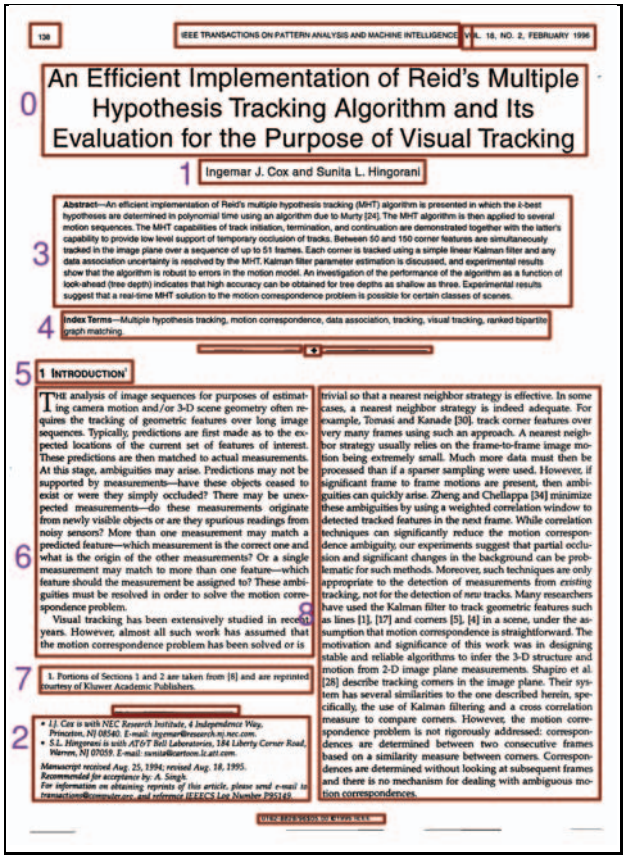

Fig. 4. A document page: the input reading order chain. Sequential numbers indicate the reading order

$X1$ and is positioned in the lower part of a page. Additional rules may state further sufficient conditions for $first\_to\_read$ and $succ\_in\_reading$.

4.3 Application of Learned Rules

Once rules have been learned, they can be applied to new documents in order to generate a set of ground atoms, such as:

$$\{first\_to\_read(0) = true, succ\_in\_reading(0,1) = true, \ldots,$$
$$succ\_in\_reading(4,3) = true, \ldots\}$$

which can be used to reconstruct chains of (possibly logically labelled) layout components. In our approach, we propose two different solutions:

1. Identification of multiple chains of layout components

2. Identification of a single chain of layout components

By applying rules learned by ATRE, it is possible to identify:

- A *directed* graph $G =< V, E >$[3] where V is the set of nodes representing all the layout components found in a document page and edges represent the existence of a *succ_in_reading* relation between two layout components, that is, $E = \{(b_1, b_2) \in V^2 | succ\_in\_reading(b_1, b_2) = true\}$
- A list of initial nodes $I = \{b \in V | first\_to\_read(b) = true\}$

Both approaches make use of G and I in order to identify chains.

Multiple Chains Identification

This approach aims at identifying a (possibly empty) set of chains over the set of logical components in the same document page. It is two-stepped. The first step aims at identifying the heads (first elements) of the possible chains, that is, the set

$$Heads = I \cup \{b_1 \in V |\ \exists b_2 \in V\ (b_1, b_2) \in E\ \wedge \forall b_0 \in V\ (b_0, b_1) \notin E\}.$$

This set contains both nodes for which $first\_to\_read$ is true and nodes which occur as a first argument in a true *succ_in_reading* atom and do not occur as a second argument in any true *succ_in_reading* atom.

Once the set $Heads$ has been identified, it is necessary to reconstruct the distinct chains. Intuitively, each chain is the list of nodes forming a path in G which begins with a node in $Heads$ and ends with a node without outgoing edges. Formally, an extracted chain $C \subseteq E$ is defined as follows:

$C = \{(b_1, b_2), (b_2, b_3), \ldots, (b_k, b_{k+1})\}$, such that

- $b_1 \in Heads$,
- $\forall i = 1..k : (b_i, b_{i+1}) \in E$ and
- $\forall b \in V\ (b_{k+1}, b) \notin E$.

In order to avoid cyclic paths, we impose that the same node cannot appear more than once in the same chain. The motivation for this constraint is that the same layout component is generally not read more than once by the reader.

Single Chain Identification

The result of the second approach is a single chain. Following the proposal reported in [28], we aim at iteratively evaluating the most promising node to be appended to the resulting chain.

More formally, let $PREF_G : V \times V \rightarrow \{0, 1\}$ be a preference function defined as follows:

[3] G is not a direct acyclic graph (dag) since it could also contain cycles

$$PREF_G(b_1, b_2) = \begin{cases} 1 \text{ if a path connecting } b_1 \text{ and } b_2 \text{ exists in } G \\ 1 \text{ if } b_1 = b_2 \\ 0 \text{ otherwise} \end{cases}$$

Let $\mu : V \rightarrow \mathbb{N}$ be the function defined as follows:

$$\mu(L, G, I, b) = countConnections(L, G, I, b) + outGoing(V/L, b) \\ -inComing(V/L, b)$$

where

- $G =< V, E >$ is the ordered graph
- L is a list of *distinct* nodes in G
- $b \in V/L$ is a candidate node
- $countConnections(L, G, I, b) = |\{d \in L \cup I | PREF_G(d, b) = 1\}|$ counts the number of nodes in $L \cup I$ from which b is reachable.
- $outGoing(V/L, b) = |\{d \in V/L | PREF_G(b, d) = 1\}|$ counts the number of nodes in V/L reachable from b.
- $inComing(V/L, b) = |\{d \in V/L | PREF_G(d, b) = 1\}|$ counts the number of nodes in V/L from which b is reachable.

Algorithm 21 fully specifies the method for the single chain identification. The rationale is that at each step a node is added to the final chain. Such a node is that for which μ is the highest. Higher values of μ are given to nodes which can be reached from I, as well as from other nodes already added to the chain, and have a high (low) number of outgoing (incoming) paths to (from) nodes in V/L. Indeed, the algorithm returns an ordered list of nodes which could be straightforwardly transformed into a chain.

5 Experiments

In order to evaluate the applicability of the proposed approach to reading order identification, we considered a set of multi-page articles published in an international journal. In particular, we considered twenty-four papers, published as either regular or short articles, in the IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), in the January and February issues of 1996. Each paper is a multi-page document; therefore, we processed 211 document images. Each document page corresponds to an RGB 24bit colour image in TIFF format.

Initially, document images are pre-processed by WISDOM++ in order to segment them, perform layout analysis, identify the membership class and map the layout structure of each page into the logical structure. Training examples are then generated by manually specifying the reading order. In all, 211 positive examples and 3,263 negative examples for the concept $fisrt\_to\_read/1$

Algorithm 1 Single chain identification algorithm

```
1: findChain (G =< V, E >, I)
   Output: L: chain of nodes
2: L← ∅;
3: repeat
4:    best_mu ← −∞;
5:    for all b ∈ V/L do
6:       cc ← countConnections(L, G, I, b);
7:       inC ← incoming(V/L, b);
8:       outG ← outGoing(V/L, b);
9:       if ((cc ≠ 0) AND (inC ≠ 0) AND (outG ≠ 0)) then
10:         μ ← cc + outG − inC;
11:         if best_mu < μ then
12:            best_b ← b;
13:            best_mu ← μ;
14:         end if
15:      end if
16:   end for
17:   if (best_mu <> −∞) then
18:      L.add(best_b);
19:   end if
20: until best_mu = −∞
21: return L
```

and 1,418 positive examples and 15,518 negative examples for the concept *succ_in_reading*/2 are generated.

We evaluated the performance of the proposed approach by means of a 6-fold cross-validation, that is, the dataset is first divided into six *folds* of near-equal size (see Table 2), and then, for every fold, the learner is trained on the remaining folds and tested on them.

When generating descriptions, the following parameters have been set: α=4, β=50 and γ=100. In the task at hand, the following logical labels are considered: *abstract*, *affiliation*, *author*, *biography*, *caption*, *figure*, *formulae*, *index_term*, *reference*, *table*, *page_no*, *paragraph*, *running_head*, *section_title*, *subsection_title*, *title*.

For each learning problem, statistics on precision and recall of the learned logical theory are recorded. In order to evaluate the ordering returned by the proposed approach, we resort to metrics used in information retrieval in order to evaluate the returned ranking of results [29]. For this purpose several metrics have been defined in the literature. Herein we consider the metrics valid for partial orders evaluation.

In particular, we consider the *normalized Spearman footrule distance* which, given two complete lists L and L_1 on a set S (that is, L and L_1 are two different permutations without repetition of all the elements in S), is defined as follows:

Fold	Article ID	Number of pages	Tot number of pages
FOLD1	tpami13	3	36
	tpami11	6	
	tpami04	14	
	tpami16	13	
FOLD2	tpami07	6	38
	tpami24	6	
	tpami17	13	
	tpami01	13	
FOLD3	tpami06	1	33
	tpami19	17	
	tpami23	7	
	tpami02	8	
FOLD4	tpami05	6	35
	tpami18	10	
	tpami22	5	
	tpami03	14	
FOLD5	tpami10	3	33
	tpami20	14	
	tpami21	11	
	tpami08	5	
FOLD5	tpami15	15	36
	tpami09	5	
	tpami14	10	
	tpami12	6	

Table 2. Processed documents

$$F(L, L_1) = \frac{\sum_{b \in S} abs(pos(L, b) - pos(L_1, b))}{|S|^2/2} \tag{1}$$

where the function $pos(L, b)$ returns the position of the element b in the ordered list L.

This measure can be straightforwardly generalized to the case of several lists:

$$F(L, L_1, \ldots, L_k) = 1/k \sum_{i=1 \ldots k} F(L, L_i). \tag{2}$$

Indeed, this measure is specifically designed for total orders and not for partial ones. In order to consider partial orders, we resorted to a variant of this measure (*induced normalized footrule distance*).

$$F(L, L_1, \ldots, L_k) = 1/k \sum_{i=1 \ldots k} F(L|_{L_i}, L_i) \tag{3}$$

where $L|_{L_i}$ is the projection of L on L_i. Since this measure does not take into account the length of single lists, we also adopted the *normalized scaled footrule distance*:

$$F'(L, L_1) = \frac{\sum_{b \in S} abs(pos(L, b)/|L| - pos(L_1, b)/|L_1|)}{|L_1|/2}. \tag{4}$$

Also in this case it is possible to extend the measure to the case of multiple lists:

$$F'(L, L_1, \ldots, L_k) = 1/k \sum_{i=1\ldots k} F'(L|_{L_i}, L_i). \quad (5)$$

In this study, we apply such distance measures to chains. In particular:

- FD=$F(L|_{L_1}, L_1)$ is used in the evaluation of single chain identification.
- SFD=$F'(L|_{L_1}, L_1)$ is used in the evaluation of single chain identification.
- IFD=$F(L, L_1, \ldots, L_k)$ is used in the evaluation of multiple chains identification.
- ISFD=$F'(L, L_1, \ldots, L_k)$ is used in the evaluation of multiple chains identification.

Results reported in Table 3 show that the system has a precision of about 65% and a recall greater than 75%. Some statistics concerning the learned theories are reported in Table 4. It is noteworthy that rules learned for the concept $first\_to\_read$ cover (on average) fewer positive examples than rules learned for the concept *succ_in_reading*. Moreover, by considering the results reported in Table 5, we note that there is no significant difference in terms of recall between the two concepts, while precision is higher for rules concerning the *succ_in_reading* concept. This is mainly due to the specificity of rules learned for the concept $first\_to\_read$ and we can conclude that the concept $first\_to\_read$ appears to be more complex to learn than the concept *succ_in_reading*. This can be explained by the limited number of training examples for this concept (one per page).

	Precision%	**Recall%**
FOLD1	76.60	61.80
FOLD2	73.00	64.90
FOLD3	80.10	67.40
FOLD4	68.00	58.20
FOLD5	76.80	68.40
FOLD6	78.20	62.60
AVG	75.45	63.88

Table 3. Overall Precision and Recall results

Concept	$first\_to\_read/1$		$succ\_in\_reading/2$	
	NOC	Training POS exs	NOC	Training POS exs
FOLD1	42	175	162	1226
FOLD2	46	173	145	1194
FOLD3	42	178	149	1141
FOLD4	42	176	114	1171
FOLD5	40	178	166	1185
FOLD6	41	175	177	1173
AVG coverage	4.17		7.77	

Table 4. Number of rules per positive examples

In the following we report some rules learned by ATRE:

Concept	$first\_to\_read/1$		$succ\_in\_reading/2$	
	Precision %	Recall%	Precision%	Recall%
FOLD1	75.00	50.00	76.90	64.10
FOLD2	66.70	63.20	74.10	65.20
FOLD3	74.30	78.80	81.00	66.10
FOLD4	69.40	71.40	67.80	56.30
FOLD5	66.70	66.70	78.40	68.70
FOLD6	71.00	61.10	79.40	62.90
AVG	70.52%	65.20%	76.27%	63.88%

Table 5. Precision and Recall results shown per concept to be learned

1. $first\_to\_read(X1) = true \leftarrow x\_pos\_centre(X1) \in [55..177]$, $y\_pos\_centre(X1) \in [60..121], height(X1) \in [98..138]$.
2. $first\_to\_read(X1) = true \leftarrow title(X1) = true$, $x\_pos\_centre(X1) \in [293..341], succ\_in\_reading(X1, X2) = true$.
3. $succ\_in\_reading(X2, X1) = true \leftarrow affiliation(X1) = true$, $author(X2) = true, height(X1) \in [45..124]$.
4. $succ\_in\_reading(X2, X1) = true \leftarrow alignment(X1, X3) = both\_columns$, $on\_top(X2, X3) = true, succ\_in\_reading(X1, X3) = true$, $height(X1) \in [10..15]$.

They can be easily interpreted. For instance, the first rule states that a block at the top of the page, horizontally positioned in the center-left part of the page with a height between 98 an 138 pixels, is the first block to read.

The second rule states that if a block represents the title, is horizontally positioned in the center of the document page and is read before another block, then it is the first to be read. This rule captures concept dependencies. In particular, the predicate $first\_to\_read$ is defined in terms of the predicate $succ\_in\_reading$.

The third rule states that a layout component whose height is between 45 and 124 pixels and labeled as 'affiliation' is read after the logical component 'author'. Since affiliation and author are not close to each other in a typical document page (see Figure 4), this rule would not have been discovered without considering results of the logical structure identification phase.

The fourth rule presents both an example of recursion on the predicate $succ\_in\_reading$ and an example of use of descriptors defined in the background knowledge ($alignment(X1, X3) = both\_columns$).

Experimental results concerning the reconstruction of single/multiple chains are reported in Table 6. We recall that the lower the distance value, the better the reconstruction of the original chain(s). By comparing results in terms of the *footrule distance* measure (IFD vs FD), we note that the reconstruction of multiple chains shows better results than the reconstruction of single chains. Indeed, this result does not take into account the length of the lists. When considering the length of the lists (ISFD vs. SFD), the situation is completely different and the reconstruction of single chains outperforms the reconstruction of multiple chains.

Concept	Multiple chains		Single chain	
	AVG. IFD%	AVG. ISFD%	AVG. FD%	AVG. SFD%
FOLD1	13.18	21.12	47.33	10.17
FOLD2	10.98	18.51	46.32	8.13
FOLD3	1.31	26.91	47.32	17.63
FOLD4	1.32	24.00	49.96	14.51
FOLD5	0.90	22.50	49.31	10.60
FOLD6	0.90	27.65	54.38	12.97
AVG	4.76%	23.45%	49.10%	12.33%

Table 6. Reading order reconstruction results

6 Conclusions

In this chapter, we present a novel approach for automatically determining the reading order in a document image understanding process. Reading order identification is a crucial problem for several applications since it permits us to reconstruct a single textual component to be used in subsequent text processing steps, such as information extraction, information retrieval and text reconstruction for rendering purposes. The proposed approach aims at learning rules which are used for predicting reading order chains of layout components detected in document images. The rules are learned from training examples consisting of sets of ordered layout components described by means of both layout and logical properties. The proposed approach presents two main peculiarities. First, it fully exploits spatial information embedded in the layout structure by resorting to inductive logic programming techniques. Second, it reconstructs reading order chains, which may not necessarily define a total ordering. This last aspect permits us to take into account the case in which independent pieces of information are represented on the same page (e.g., the end of an article and the beginning of a new one) and the case in which some layout components should not be included in the reading order (e.g. images or page numbers).

In the learning phase, rules which identify the first logical component to read and define the successor relation are induced. In the recognition phase such rules are used to reconstruct reading order chains according two different modalities: single vs. multiple chains identification. Results prove that learned rules are quite accurate and that the reconstruction phase significantly depends on the application at hand. In particular, if the user is interested in reconstructing the actual chain (e.g. text reconstruction for rendering purposes), the best solution is in the identification of single chains. On the contrary, when the user is interested in recomposing a text such that sequential components are correctly linked (e.g. in information extraction applications), the most promising solution is the identification of multiple chains.

For future work we intend to consider the entire document (and not the single page) as the analysis unit. This would permit us to reconstruct multiple crossing-pages chains typically found in collections of documents (e.g., conference proceedings or transcriptions of ancient edicts).

References

1. Nagy, G., Seth, S., Viswanathan, M.: A prototype document image analysis system for technical journals. Computer **25**(7) (1992) 10–22
2. Dengel, A., Bleisinger, R., Hoch, R., Fein, F., Hones, F.: From paper to office document standard representation. IEEE Computer **25**(7) (1992) 63–67
3. Wenzel, C., Maus, H.: Leveraging corporate context within knowledge-based document analysis and understanding. IJDAR **3**(4) (2001) 248–260
4. Ceci, M., Berardi, M., Malerba, D.: Relational data mining and ILP for document image understanding. Applied Artificial Intelligence **21**(4-5) (2007) 317–342
5. Tsujimoto, S., Asada, H.: Understanding multi-articled documents. In: in Proceedings of the 10th International Conference on Pattern Recognition. (1990) 551–556
6. Ishitani, Y.: Document transformation system from papers to xml data based on pivot xml document method. In: ICDAR '03: Proceedings of the Seventh International Conference on Document Analysis and Recognition, Washington, DC, USA, IEEE Computer Society (2003) 250
7. Nagy, G., Seth, S.: Hierarchical representation of optically scanned documents. In: Seventh Int'l Conf. Pattern Recognition, IEEE CS Press (1984) 347–349
8. Meunier, J.L.: Optimized xy-cut for determining a page reading order. In: ICDAR '05: Proceedings of the Eighth International Conference on Document Analysis and Recognition, Washington, DC, USA, IEEE Computer Society (2005) 347–351
9. Taylor, S.L., Dahl, D.A., Lipshutz, M., Weir, C., Norton, L.M., Nilson, R., Linebarger, M.: Integrated text and image understanding for document understanding. In: HLT '94: Proceedings of the workshop on Human Language Technology, Morristown, NJ, USA, Association for Computational Linguistics (1994) 421–426
10. Aiello, M., Monz, C., Todoran, L., Worring, M.: Document understanding for a broad class of documents. International Journal on Document Analysis and Recognition-IJDAR **5**(1) (2002) 1–16
11. Allen, J.F.: Maintaining knowledge about temporal intervals. Commun. ACM **26**(11) (1983) 832–843
12. Aiello, M., Smeulders, A.M.W.: Thick 2d relations for document understanding. Inf. Sci. Inf. Comput. Sci. **167**(1-4) (2004) 147–176
13. Breuel, T.M.: High performance document layout analysis. In: Proceedings of the 2003 Symposium on Document Image Understanding (SDIUT '03). (2003)
14. Altamura, O., Esposito, F., Malerba, D.: Transforming paper documents into XML format with WISDOM++. IJDAR **4**(1) (2001) 2–17
15. Malerba, D., Esposito, F., Altamura, O., Ceci, M., Berardi, M.: Correcting the document layout: A machine learning approach. In: ICDAR '03: Proceedings of the Seventh International Conference on Document Analysis and Recognition, Washington, DC, USA, IEEE Computer Society (2003) 97–101
16. Esposito, F., Malerba, D., Lisi, F.A.: Machine learning for intelligent processing of printed documents. J. Intell. Inf. Syst. **14**(2-3) (2000) 175–198
17. Malerba, D., Esposito, F., Lisi, F.A., Altamura, O.: Automated discovery of dependencies between logical components in document image understanding. ICDAR '01: Proceedings of the Sixth International Conference on Document

Analysis and Recognition, Washington, DC, USA, IEEE Computer Society (2001) 174–178
18. Utgoff, P.: An improved algorithm for incremental induction of decision trees. In: Proc. of the Eleventh Int. Conf. on Machine Learning, Morgan Kaufmann (1994)
19. Malerba, D.: Learning recursive theories in the normal ilp setting. Fundamenta Informaticae **57**(1) (2003) 39–77
20. Grimaldi, R.P.: Discrete and Combinatorial Mathematics, an Applied Introduction. Addison Wesley, third edition (1994)
21. Muggleton, S.: Inductive Logic Programming. Academic Press, London (1992)
22. De Raedt, L.: Interactive Theory Revision. Academic Press, London (1992)
23. Lavrač, N., Džeroski, S.: Inductive Logic Programming: techniques and applications. Ellis Horwood, Chichester (1994)
24. Bergadano, F., Gunetti, D.: Inductive Logic Programming: from machine learning to software engineering. The MIT Press, Cambridge, MA (1996)
25. Nienhuys-Cheng, S.W., de Wolf, R.: Foundations of inductive logic programming. Springer, Heidelberg (1997)
26. Levi, G., Sirovich, F.: Generalized and/or graphs. Artif. Intell. **7**(3) (1976) 243–259
27. Mladenić, D., Grobelnik, M.: Feature selection for unbalanced class distribution and naive bayes. In: ICML '99: Proceedings of the Sixteenth International Conference on Machine Learning, Morgan Kaufmann Publishers Inc. (1999) 258–267
28. Cohen, W.W., Schapire, R.E., Singer, Y.: Learning to order things. J. Artif. Intell. Res. (JAIR) **10** (1999) 243–270
29. Dwork, C., Kumar, R., Naor, M., Sivakumar, D.: Rank aggregation methods for the web. In: WWW '01: Proceedings of the 10th international conference on World Wide Web, New York, NY, USA, ACM Press (2001) 613–622