An Abstract Argumentation-based Strategy for Reading Order Detection

Stefano Ferilli[1,2] and Andrea Pazienza[1]

[1] Dipartimento di Informatica – Università di Bari
{stefano.ferilli,andrea.pazienza}@uniba.it
[2] Centro Interdipartimentale per la Logica e sue Applicazioni – Università di Bari

Abstract. Document Image Analysis is the branch of Document Processing in charge of extracting high-level information from the purely pictorial appearance of the document. In between single-column documents, in which no reading order ambiguity is present, and fully partitioned documents, in which each layout component is self-contained and hence can be read independently of all the others, there are tricky cases in which (some of) the layout components in a document page are related to each other, in that they contain different portions of a single discourse (e.g., newspapers). In these cases, automatic procedures for the acquisition of the document content might be ineffective, inapplicable, or lead to inconsistent results. This paper proposes an automatic strategy for identifying the correct reading order of a document page's components based on abstract argumentation. The technique is unsupervised, and works on any kind of document based only on general assumptions about how humans behave when reading documents. Experimental results show that it is very effective, also compared to previous solutions that have been proposed in the literature.

1 Introduction

While today most documents are generated, stored and exchanged in a digital format, the typing convention of classical *paper* documents are still in use. Moreover, large amounts of *legacy* documents are digitized and stored in various kinds of repositories. To be automatically handled and managed, the content of these documents needs to be extracted and properly organized, and the huge amount of material makes this activity manually unfeasible. Hence, the need for automatic pre-processing techniques that carry out this task. The overall process a digital document typically includes three phases: Layout Analysis, Document Image Understanding and Document Understanding. *Layout Analysis* consists in the perceptual organization process that aims at identifying the single blocks of a document and at detecting geometrical relations among them (*Layout Structure*); then, extracting semantic information from this structure is the task of *Document Image Understanding*, that yields the *Document Logical Structure*. Finally, *Document Understanding* is in charge of extracting useful information from the document content. In particular, *Document Image Analysis*

(DIA) [15], that encompasses the first two phases, is the branch of Automatic Document Processing that aims at extracting high-level information from the low-level representation of a document.

The textual content of the single logical components of a document can be read after Document Image Understanding is carried out, and provided to Document Understanding. While such an extraction often does not pose any problem, because the aim is just displaying each component separately or because most documents involve a linear flow of text, in other cases the document layout is quite complex, requiring suitable strategies to determine the correct reading order of these components. Since the various articles are composed in the page in several unpredictable combinations, and have different size and number of components, a simple top-down, left-to-right reading order of the pages would be ineffective, returning a text flow that interleaves components from different unrelated articles. Hence, subsequent Document Understanding steps would be inapplicable, or would return nonsense results. So, Reading Order Detection in a document is a hot problem and new approaches are needed to tackle difficult cases, in order to provide general a flexible solutions to this problem.

This paper proposes the use of an abstract argumentation framework to solve this problem. Specifically, the proposed approach uses a representation of the problem that is totally general and applicable to any kind of document. Advantages of this solution include the fact that it does not need any (supervised or unsupervised) learning, and hence is directly applicable to any document page and complies with an incremental extension of the document base. The technique has been implemented and embedded in the DIA step of DoMInUS, a system for document processing and management that provides the Layout Analysis pre-processing techniques required to obtain the representation to be fed to the argumentation reasoner. The next section discusses related work on this topic. Then, Section 3 recalls the architecture and the main components of DoMInUS that carry out the Layout Analysis task and Section 4 summarizes the main concepts of a formal argumentation framework. Section 5 describes the solution and experimental results showing its effectiveness. Finally, Section 6 concludes the paper and outlines future work directions.

2 Related Work

The reading order detection problem consists in finding a sequence of the objects in a document's page that reflects the human reading order. Multiple approaches to this problem have been proposed in the literature. Some are based on layout information only, others exploit the content of objects, with or without using *a priori* knowledge about the particular document class.

Many works are based on the well-known XY-cuts segmentation algorithm [16] (see next section for details). Ishitani [13] proposes to exploit the hierarchy of blocks generated by the XY-cuts algorithm (called the 'XY-tree') to induce the reading order among the final blocks. However, exploiting XY-cuts to determine reading order often causes problems. Indeed, XY-cuts cannot deal with blocks

organized in L-like shapes, and it needs to know the minimum required width of the horizontal/vertical stripes for the cutting strategy. Overall, XY-cuts approaches are simple and require only visual information. However, due to their cut strategies, they perform reasonably well only on documents with simple layouts (e.g., Manhattan layouts). This is a serious limitation of these approaches.

Knowledge-based approaches use rules to identify a reading order in the pages. Typically the rules provided to the system encode general criteria concerning reading order, and are manually written by experts. In [5] geometric and linguistic information is used to determine a reading order. The key idea is to compare all possible pairs of text lines, introducing whenever appropriate suitable constraints on their reading order, derived from their geometric arrangement or of their linguistic content. In [18] *tab-stops* detection is used to deduce the columns of the page, then this information is exploited to detect blocks and determine the reading order. Three types of blocks are used to formulate five ordering rules that determine the reading order of the page.

Differently from previous approaches, Aiello et al. [3] use Natural Language Processing (NLP) techniques to improve reading order detection. For each page, rectangular components (*document-objects*) are extracted (for non-rectangular shapes the bounding box is considered) and described with geometrical and content-based features. To detect the reading order, they define a partial order relation, called *BeforeInReading*, on pairs of document-objects.

A more sophisticated approach that uses only visual information to reading order is provided in [14]. They formulate the reading order problem as a learning problem where the goal is to find a First-Order Logic (FOL) theory that is complete and consistent with respect to all training examples. Initially, they use a knowledge-based document image processing system (WISDOM++) to extract the logical structure of the page and identify the membership class of the document. Then, they use an Inductive Logic Programming system (ATRE) to learn concepts describing the reading order chains of a single document page.

It is worth noting that a common assumption, made by many approaches, concerns the uniqueness of reading order for each page. Indeed, this assumption is clearly wrong for the pages of newspapers or magazines, where many mutually independent articles are present in each page and can be read in any order. To the best of our knowledge, only [14] and [12] explicitly address the problem with the identification of multiple reading chains.

3 DoMInUS

Extending previous research presented in [9] DoMInUS (DOcument Management INtelligent Universal System) [8, 10] is a document processing and management system characterized by the intensive exploitation of intelligent techniques in each step of document processing from acquisition to indexing, from categorization to storing and retrieval. Since it is general and flexible, it can be embedded as a document management engine into many different Digital Library systems. Based on the ODA/ODIF standard, any document can be progressively parti-

tioned into a hierarchy of abstract representations, called its *layout structure*. Here we describe an approach implemented for discovering a full layout hierarchy in digital documents based primarily on layout information. DoMInUS embeds several techniques that allow it to extract the high-level geometrical structure of a document, both for born-digital documents and for digitized documents, both for Manhattan and for non-Manhattan cases:

- XY-cuts, used for digitized documents, but modifiable to work on born-digital documents as well, useful for Manhattan layout;
- Run-Length Smoothing Algorithm (RLSA), used for digitized documents only, useful for Manhattan layout;
- Run-Length Smoothing with OR (RLSO) [11], in its two versions for digitized and born-digital documents, useful for non-Manhattan layout;
- Background Structure Analysis, used for born-digital documents only, useful for Manhattan layout, but adaptable for non-Manhattan layout.

For digitized documents, the input to such techniques is the raster image for each page after pre-processing aimed at noise removal, dewarping and deskewing. For born-digital documents, in PostScript (PS) [1] or PDF [2] formats, the input to such techniques is a vectorial description of each document page in terms of *blocks*, each of which may be (a fragment of) a text line, a graphical (horizontal/vertical) line, a closed area filled with one color or a raster image. In both cases, the output is a set of *frames*, defined as collections of basic blocks or pixels completely surrounded by white space that should correspond to logical components that may be associated to a well-defined role in the document. The horizontal/vertical size and position in the page, and type of content, of the frames is reported, and used to infer various kinds of higher-level spatial relationships among frames (expressing horizontal/vertical adjacency and horizontal/vertical alignment) [17, 7]. Figure 1 shows a digital document and the corresponding output of the proposed algorithm, with the discovered frames highlighted.

4 Formal Argumentation Basics

Everyday, people must make decisions about conflicting information. For example, when a judge has to decide whether a person is guilty or not, he must base his decision on contradictory information. Indeed, the claims of the involved persons are typically contradictory. Making decisions in these situations is not trivial. The usual strategy consists in proving the inconsistency of some assertions and then finding a consistent way to link the remaining information. This is the task of *argumentation*, an inferential strategy that provides a general approach to model defeasible and non-monotonic reasoning. Over the last years, it has become an influential subfield of Artificial Intelligence, with applications ranging from legal reasoning, to dialogues and persuasion, to medicine, to eGovernment.

A foundational work for the abstract argumentation theory was proposed by Dung [6]. According to his setting, an *abstract argument system* or *Argumentation Framework* (AF for short) is a pair $\langle A, R\rangle$ consisting of a set A, whose

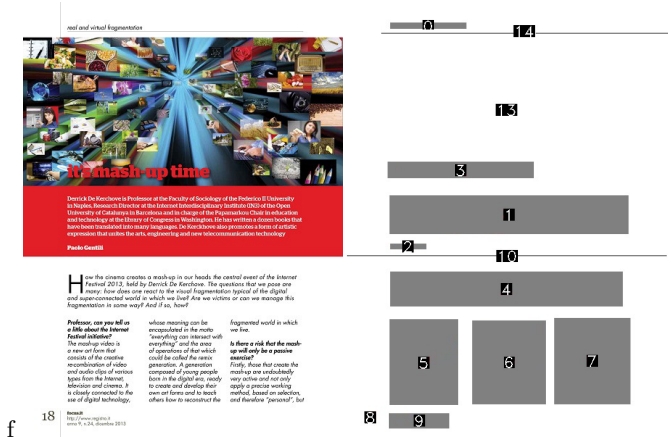

Fig. 1. Layout structure extracted by DoMInUS on a sample document

elements are called *arguments*, and a binary relation $R \subseteq A \times A$, called *attack relation*. Given two arguments $\alpha, \beta \in A$, the relation $\alpha R \beta$ represents an attack from α against β. In general, arguments α and β are in conflict if argument α refutes argument β or if α is attacking premises supporting β. This setting can be represented as a directed graph, with each node representing an argument and each edge representing an attack. Given such a graph, the objective is determining which subset(s) of its nodes can be justified.

A basic requirement in this setting is the concept of *admissibility*. A set $S \subseteq A$ of arguments is admissible if it is *conflict-free* (i.e., $\nexists \alpha, \beta \in S$ s.t. $\alpha R \beta$), and *acceptable* (i.e., $\forall \beta \in A : \beta R \alpha \Rightarrow \exists \gamma \in S$ s.t. $\gamma R \beta$). For example, if $\{A, B, C, D, E\}$ is included in a set of arguments and the attacks are defined as shown in Figure 2, then the subset $S = \{B, D\}$ (double circles in the figure) is admissible because it defends all its arguments (i.e., it is acceptable) and there not exists any attack between its arguments (i.e., it is conflict-free). As observed in [4], in an AF one is usually interested in the *justification state* of the arguments. The justification state in an AF can be determined according to suitable *semantics*, that specify how to derive from an AF a set of extensions that intuitively represents a set of arguments which are collectively justified.

Definition 1 (semantics). *Let $AF = \langle A, R \rangle$ be an Argumentation Framework, $S \subseteq A$ be a conflict-free set of arguments and $F : 2^A \rightarrow 2^A$ be defined as $F(S) = \{a \mid a$ is defended by $S\}$. Then,*

- *S is a* Complete Extension *iff $S = F(S)$;*
- *S is a* Grounded Extension *iff S is the minimal Complete Extension (w.r.t. set inclusion);*
- *S is a* Preferred Extension *iff S is the maximal Complete Extension (w.r.t. set inclusion);*
- *S is a* Stable Extension *iff S is a Complete Extension that attacks every argument in $A \setminus S$;*

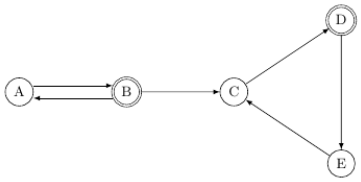

Fig. 2. Graph representation of an AF. Double circles represent an acceptable set

Each semantics involves a different degrees of skepticism that can be used to evaluate the arguments. So, a partial order relation can be established among semantics. Among the semantics in Definition 1, the most skeptical is the Grounded Extension, while the most credulous the Preferred Extension. Preferred and Ground Extensions are also Complete. Considering again the example in Figure 2, we have that $\{\emptyset, \{A\}, \{B, D\}\}$ are Complete Extensions, $\{\{A\}, \{B, D\}\}$ are Preferred Extensions, $\{B, D\}$ is a Stable Extension, and $\emptyset$ is the only Ground Extension.

5 Argumentation-based Reading Order Detection

Summing up, we aim at designing a procedure for reading order detection in document pages that fulfills several requirements ensuring maximum generality and widest applicability. First of all, the procedure must not take into account the textual content of the document blocks, but just their layout organization (hence, the exploitation of NLP techniques is not required). Indeed, using the textual content would require the exploitation of NLP techniques, and it is well-known that these technique are language-dependent and are not available for all languages, or at least not with the same quality. Conversely, our proposed approach can be applicable to any kind of document because it relies basically on the position of components resulting from the layout structure of the page(s).

This is consistent with the human approach, in which the reading order is determined without actually reading the document[3]. Also, our technique is specifically interested in non-Manhattan layouts, where techniques based on a partition of the page according to its background are not applicable. For instance, newspapers are a good representative of this kind of documents. Third, we want the technique to be based on very simple layout information, that can be easily and reliably extracted from an automatic procedure and is independent of the specific kind of document. Having such a low-level input clearly places most burden on the reading order detection technique. While this complexity is often tackled using knowledge-based approaches, we want to avoid this setting, because

[3] It may be necessary to leverage the textual content in order to solve inconsistencies or ambiguities in the text flow across subsequent components in the determined reading order. E.g., in Figure 3 one would expect the horizontally-oriented reading order (0,1,2,3,4,5), while the correct one is vertically-oriented (0,3,1,4,2,5).

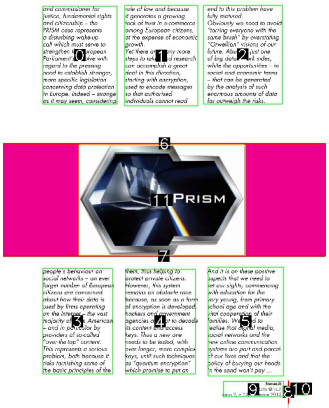

Fig. 3. Sample document having a counter-intuitive reading order

hand-written rules are costly, error-prone and typically depend on the kind of document. We want to avoid also the Machine Learning-based solution to this problem, by which such rules are automatically obtained by the system starting from examples of manually labeled documents. Indeed, a manual intervention is anyway required, and the learned knowledge is in any case dependent on the kind of documents, and on the specific documents, used for training the system. So, it is valid only up to proof of the contrary. Finally, it is often the case that the reading order in a document page does not determine a total order, but the components can be partitioned into independent subgroups, each characterized by its own reading order, leaving the reader free to decide an order among groups.

Among the various inferential strategies available in Artificial Intelligence, Abstract Argumentation seemed to provide the proper tools to deal with all of the above requirements. Indeed, one might just express possible (even trivial) partial reading orderings and identify pairs of these partial solutions that are mutually inconsistent. Given a document page image, we run DoMInUS to obtain the layout structure, consisting of layout blocks labeled with their type of content. For the reading order detection purposes, image blocks are simply ignored. Moreover, horizontal or vertical lines are considered as natural separators when their projection spans more than one content (i.e., image or text) block. The presence of these separators allows to partition the page into independent portions in which the reading order can be determined separately, which slightly simplifies the problem. Frames in a page can be considered as made up of 4 different lines to which this perspective can be applied. Given the text blocks in the (portion of) page under processing, we consider only the following very basic and document-independent reading rules for providing the input to our technique:

– horizontally or vertically adjacent components are candidates to be read consequently;

– a component at the bottom of the (portion of) page might be followed by a component at the top of an adjacent column, and
– a rightmost (resp., leftmost) component might be followed by a leftmost (resp., rightmost) component in an adjacent row.

As required, these rules are so trivial and general that a computer may easily identify, given the document layout, which pairs of components fulfill them. Each pair of blocks (A, B) in the considered (portion of) document that fulfills any of these requirements is translated into an argument representing the claim "components A and B are to be read one after the other in a document". Formally, we express this in FOL using predicate `next/2`, as `next(`A`,`B`)`. Note that this predicate does not imply any direction in the relationship between A and B, and hence it applies both to languages in which the reading order proceeds left-to-right and to those in which it proceeds right-to-left.

Arguments of this kind are so simple and intuitive that it is also immediate to automatically infer the possible attacks to be provided to the argumentation engine. Indeed, given three components A, B and C, one knows that arguments of the type `next(`A`,`B`)` and `next(`A`,`C`)` mutually attack each other, because if B is to be read after A, then C cannot be read after A; conversely, if C is to be read after A, then B cannot be read after A. This can be expressed by the following attacks in our Argumentation Framework:

`attacks(next(`A`,`B`),next(`A`,`C`)), attacks(next(`A`,`C`),next(`A`,`B`)).`

The same holds for two arguments of the type `next(`A`,`C`)` and `next(`B`,`C`)`: if A is to be read immediately before C, then B cannot be read immediately before C as well, and *vice-versa*:

`attacks(next(`A`,`C`),next(`B`,`C`)), attacks(next(`B`,`C`),next(`A`,`C`)).`

For instance, the document page in Figure 1 yields the following formal description, corresponding to the segments in Figure 4a:

(0,3), (1,2), (3,1), (4,5), (4,6), (4,7), (5,6), (5,9), (6,7), (8,0), (8,4), (8,9)

for which the following attacks are automatically derived:

(4,5)-(4,6), (4,6)-(4,5), (4,6)-(4,7), (4,7)-(4,6), (4,5)-(4,7), (4,7)-(4,5),
(5,6)-(5,9), (5,9)-(5,6), (8,0)-(8,4), (8,4)-(8,0), (8,0)-(8,9), (8,9)-(8,0),
(8,4)-(8,9), (8,9)-(8,4), (4,6)-(5,6), (5,6)-(4,6), (4,7)-(6,7), (6,7)-(4,7),
(5,9)-(8,9), (8,9)-(5,9)

The correct reading order is the following (graphically shown in Figure 4b):

(3,1), (1,2), (2,4), (4,5), (5,6), (6,7)

Our technique returned exactly this reading order, except for relation (2,4). This is due to the presence of a ruling line between the paper heading and its body. Actually, this may be considered acceptable, since the heading and the body may indeed be read independently.

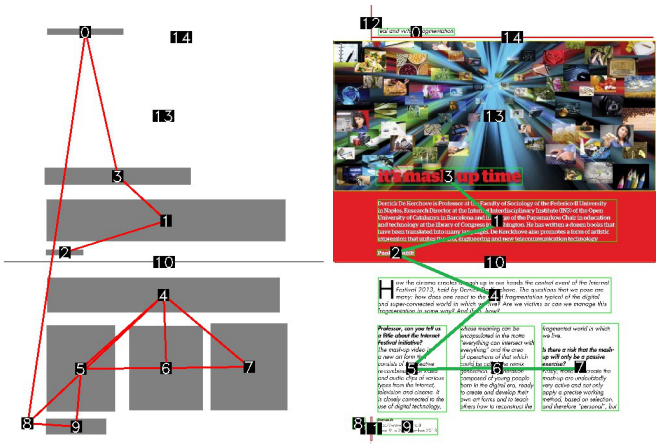

Fig. 4. Correct reading order in a document description

The proposed technique was tested on a dataset including 103 document pages of different layout complexity, taken from newspapers, magazines and (scientific) papers. Trivial cases of single-column documents were not included in the dataset, while tricky cases (such as the one in Figure 3) are present. Statistics about the dataset are reported in the top and middle rows of Table 1, both for single classes and for the whole dataset (in the last column). Table cells report average values, with minimum-maximum range in parentheses. The last column reports both the averages computed on single documents, and the averages over class averages (in parentheses). The dataset involved on average 20.73 blocks, 22.98 arguments and 63.17 attacks per document. It is evident that, as expected, the simplest class is 'Magazine', while the most complex one is 'Newspaper', both for the number of involved components and for the reading order-related relationships. Figure 5 shows the structure of the most complex document in the dataset, a newspaper page involving 106 blocks, 137 arguments and 1020 attacks. The last two rows of Table 1 report the figures of the experimental results, using the same representation as for the dataset statistics. The justified set of arguments was determined using Preferred Extensions. Interestingly, for this parameter the 'Magazine' and 'Paper' classes become very close to each other. While the maximum number of Preferred Extensions is still higher for papers than for magazines, on average 'Magazine' turns out to have slightly more Preferred Extensions. However, the gap between these classes and 'Newspaper' is huge (the number of Preferred Extensions in the latter is 3 orders of magnitude larger than in the former). This is due to the fact that pages with complex layout arrangement are often partitioned so that reading order is relevant for blocks within the same element of the partition, but reading order of partition elements is independent. Newspaper pages in the dataset have a significant impact on the overall complexity of the dataset, as can be noted by looking at the averages in the last column.

Fig. 5. A very complex document in the dataset

For each extension, the recall was evaluated as the ratio of correct `next/2` items retrieved over `next/2` items in the correct order sequence. Since a page may admit several Preferred Extensions (i.e., alternative correct reading orders), the recall of a given page was determined as the average recall over all Preferred Extensions for that page. The proposed technique reached 77.94% average recall on the entire dataset (79.73% considering the compound average over classes). Consistently with the class complexity, the worst recall occurred on newspapers, but still being quite satisfactory (70.74%) for such a difficult class. Interestingly, the best recall was reached on papers (91.32%), even if they have a more complex structure than magazines. Conversely, the performance on magazines is very close to that for newspapers (71.75%), despite its much less complex structure. Since the competitor systems, and/or the dataset on which they were run, are not available, we have run our own experiments and compared our performance with those reported in the other papers. It should be noted that we included in our dataset very complex cases having non-Manhattan layout. Our performance is comparable to that of previous systems, which can be considered a success, since we take a trivial input that does not require high-level interpretation. and we can deal with very complex non-Manhattan layouts. As regards techniques that handle many possible reading orders, we are better than [14]. We are worse than [12], but [12] uses linguistic information, which is not always available, while we can deal with any kind of document independently of the language in which it is written.

We also carried out a qualitative analysis to spot problems and get hints for improvement. Most problems are due to the fact that sometimes there are fancy reading orders that deviate from the general 'top-down, left-to-right' rule. These are the cases where linguistic information may help. This is an intrinsic limitation of our approach, that we accepted when we ruled out any language-based processing. Other minor problems concerned the separation of the portions of the

Table 1. Dataset and experimental statistics.

	Paper	Magazine	Newspaper	Overall
#Documents	43	40	20	103
Text blocks	15.18 (5-14)	8.13 (5-13)	26.35 (16-66)	14.61 (16.55)
Image blocks	1.05 (0-6)	0.70 (0-1)	5.35 (3-11)	1.75 (2.37)
Separators (lines)	0.37 (0-3)	2.60 (0-5)	16.50 (8-50)	4.37 (6.49)
#Blocks	16.60 (3-27)	11.42 (7-18)	48.20 (29-106)	20.73 (25.41)
#Arguments	24.13 (2-42)	9.20 (3-19)	44.45 (21-137)	22.98 (25.93)
#Attacks	61.67 (0-180)	14.20 (0-62)	164.3 (42-1020)	63.17 (80.06)
#Preferred extensions	6.65 (1-120)	6.83 (1-108)	5 053.85 (14-59 916)	986.76 (1 689.11)
Recall(%)	91.32 (62.50-100)	71.75 (0-100)	70.74 (10.80-100)	77.94 (79.73)

pages. On papers, the technique fails when header and footer are not separated by lines; on magazines, when multi-line titles are across different backgrounds; on newspapers when columns of the same article are separated by lines. These problems might be tackled by refining the rules to consider additional layout information, such as spacing and font size.

6 Conclusions

DIA aims at automatically extracting high-level information from the purely pictorial appearance of the document. A task of DIA is determining the reading order among text components in a document page. This is a required step to ensure applicability and effectiveness of automatic procedures for the acquisition of the document content. While this may be trivial in single-column documents or in documents in which each layout component is self-contained, there are tricky cases in which (some of) the layout components in a document page are related to each other, in that they contain different portions of a single discourse.

This paper proposed an automatic strategy for identifying the correct reading order of a document page's components based on abstract argumentation. The technique is unsupervised, and works on any kind of document based only on general assumptions about how humans behave when reading documents. Experimental results show that it is very effective, also compared to previous solutions that have been proposed in the literature. Qualitative analysis of the results suggested possible directions for further improvement of the approach.

Acknowledgments

This work was partially funded by the Italian PON 2007-2013 project PON02_00563_3489339 'Puglia@Service'.

References

[1] Adobe Systems Inc. *PostScript language reference manual – 2nd ed.* Addison Wesley, 1990.

[2] Adobe Systems Inc. *PDF Reference version 1.3 – 2nd ed.* Addison Wesley, 2000.

[3] M. Aiello, C. Monz, and L. Todoran. Document understanding for a broad class of documents. *IJDAR*, 5(1):1–16, 2002.

[4] P. Baroni and M. Giacomin. Semantics of abstract argument systems. In Guillermo Simari and Iyad Rahwan, editors, *Argumentation in Artificial Intelligence*, pages 25–44. Springer US, 2009.

[5] Thomas M Breuel. High performance document layout analysis. In *Proc. Symp. Document Image Understanding Technology*, 2003.

[6] P.M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artif. Intell.*, 77(2):321–358, 1995.

[7] M. Egenhofer. Reasoning about binary topological relations. In O. Gunther and H.-J. Schek, editors, *Second Symposium on Large Spatial Databases*, volume 525 of *LNCS*, pages 143–160. Springer, 1991.

[8] F. Esposito, S. Ferilli, T.M.A. Basile, and N. Di Mauro. Machine learning for digital document processing: From layout analysis to metadata extraction. In S. Marinai and H. Fujisawa, editors, *Machine Learning in Document Analysis and Recognition*, volume 90 of *Studies in Computational Intelligence*, pages 79–112. Springer, 2008.

[9] F. Esposito, D. Malerba, G. Semeraro, S. Ferilli, O. Altamura, T.M.A. Basile, M. Berardi, M. Ceci, and N. Di Mauro. Machine learning methods for automatically processing historical documents: From paper acquisition to XML transformation. In *DIAL 2004*, pages 328–335, 2004.

[10] S. Ferilli. *Automatic Digital Document Processing and Management - Problems, Algorithms and Techniques.* Adv. in Pattern Recognition series. Springer, 2011.

[11] S. Ferilli, M. Biba, F. Esposito, and T.M.A. Basile. A distance-based technique for non-manhattan layout analysis. In *ICDAR-2009*, volume I, pages 231–235. IEEE Computer Society, 2009.

[12] L. Gao, Z. Tang, X. Lin, and Y. Wang. A graph-based method of newspaper article reconstruction. In *ICPR*, pages 1566–1569, 2012.

[13] Y. Ishitani. Document transformation system from papers to xml data based on pivot xml document method. In *Document Analysis and Recognition, 2003. Proceedings. Seventh International Conference on*, pages 250–255 vol.1, Aug 2003.

[14] D. Malerba, M. Ceci, and M. Berardi. Machine learning for reading order detection in document image understanding. In S.. Marinai and H. Fujisawa, editors, *Machine Learning in Document Analysis and Recognition*, volume 90 of *Studies in Computational Intelligence*, pages 45–69. Springer, 2008.

[15] G. Nagy. Twenty years of document image analysis in PAMI. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):38–62, 2000.

[16] G. Nagy and S.C. Seth. Hierarchical representation of optically scanned documents. In *ICPR*, pages 347–349. IEEE, 1984.

[17] D. Papadias and Y. Theodoridis. Spatial relations, minimum bounding rectangles, and spatial data structures. *International Journal of Geographical Information Science*, 11(2):111–138, 1997.

[18] R. Smith. Hybrid page layout analysis via tab-stop detection. In *Document Analysis and Recognition, 2009. ICDAR '09. 10th International Conference on*, pages 241–245. IEEE, July 2009.