

# Unsupervised Method to Generate Page Templates

Hervé Déjean  
Xerox Research Centre Europe  
Herve.Dejean@xrce.xerox.com

## ABSTRACT

In this paper, we propose a method for automatically inferring the different page templates used to layout the document content. The first step of the method consists in performing a logical analysis of the document. Depending of the coverage of this step, a given number of document elements will be labeled. Then geometric relations are computed between these labeled elements, and page templates candidates are generated using frequent related elements. A fuzzy matching operation allows for selecting the most frequent and relevant page templates for a given document. Such page templates can be used to correct errors produced during the different previous steps of the document analysis: zoning, OCR, and logical analysis. Evaluation has been performed using the INEX book track collection.

**Keywords:** document layout analysis, geometrical analysis, logical analysis, page template, typography, unsupervised learning

## 1. INTRODUCTION

Document Analysis is often described as a two-step approach: first a physical/geometrical layout analysis is performed, whose goal is to identify a set of homogeneous regions of a given type, mainly text and figure. The next step, a logical layout analysis, aims at labeling the regions with a specific label. The methods used for this labeling task very often rely on *a priori* knowledge of the type of documents. As mentioned in [1], the output characterization of both steps is not particularly well-defined: the terms *page model* and *document model* refer to different formalizations and scope in the literature. Following [1], we consider that a *better view* [sic] would be to model these outputs in terms of typesetting systems. We present here a first experiment around the notion of *page template*, commonly used by typesetting systems to organize the global layout of a document. We present in this paper a method to automatically infer such page templates from a document without manual annotation.

The use of page layout model is not new in the field of Document Analysis. One application relates to document/page categorization and/or retrieval [10;12]. A more related work is around logical labeling: elements are labeled using local features but also spatial/geometrical relations between elements in a page [2;6;7]. This work is usually illustrated or evaluated using very specific and highly-structured samples such as business cards or the title pages of articles. An interesting work is presented in [6;7]: a graph-based approach is used to label elements in a page. Their *model layout graph* is *a priori* known and annotated examples are required to tune the system. In [12], a similar representation of document layout is used, but the layout graph is inferred from labeled samples (elements are manually labeled). In this paper, instead of providing specific page templates, logical labeling components are required to bootstrap the template inference

Most of the types of documents used for these experiments correspond to pages with many labeled element such as title pages, business cards, letters. In this work we would like to show that the use of page template is worthy even for "common" books such as novels, and that the automatic inference of these page templates is possible, given a set of generic components. We will show that a main advantage of the method is to be able to correct labeling errors made by these components. The remaining part of the article is organized as follows: Section 2 details the notion of page template and illustrates it with an example. We use simple but relevant typographical notions in order to ground the model of page template. Then Section 3 describes the different steps of the method, and evaluation is provided Section 4. We conclude by explaining the different usages of the inferred page templates in a Document Conversion system.

## 2. NOTION OF PAGE TEMPLATE

Notions such as document/page template and master document/page are common in any typesetting system (even though they do not cover exactly the same notion from system to system). OpenOffice Writer uses *page style* to describe pages

in a document: "In Writer, page styles define the basic layout of pages, including page size, margins, headers and footers, borders and backgrounds, number of columns, and so on.". [OpenOffice wiki, working with page styles]

This work uses the fact that document content is laid out using a set of page templates. Reverse engineering from a given document should then be possible in order to infer its templates. Our work aims then at automatically *inferring* a set of page templates after a logical analysis has been performed on a given document. Our page layout meta-template definition is somehow similar to these found in [4;6] and is the following: a page template is composed of a set of labeled elements having typography-based geometric relations. Formally a page template is a graph, where nodes correspond to labeled elements and edges correspond to geometric relations,

$$\text{pageTemplate} = \{ \{n_i\}_{i=1..N}, \{R(n_i, n_j)_{i,j=1,2..M}\} \}$$

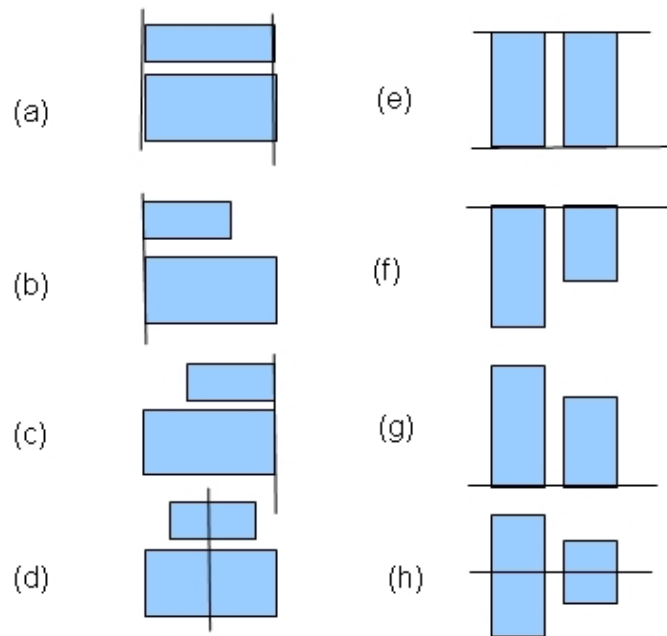
$n_i$  = (label), where the labels correspond to the labeling components.

$$R(a,b) = \{L(a,b), \text{value}\}$$

$$L \in (\{I\text{-justified}, I\text{-centered}, I\text{-flush-left}, I\text{-flush-right}, I \text{ in } \{\text{vertical}, \text{horizontal}\}\},$$

where  $N$  is the number of nodes (labeled elements), and  $M$  the number of relations between nodes.

Elements are commonly represented by zones (rectangles). They are **labeled**: Only elements recognized by the logical analysis step are considered by our method. Our logical component library covers the following elements: page header and footer, page body, enumerated pattern, page number, table of contents entry and corresponding title in body, caption, footnote, drop cap, image. The different relations between 2 elements are shown below. Since we attempt to model page template in terms of typesetting system, we use common *typographical* alignments:



**Figure 1: Typographical relations used: justified (a, e), right and left ragged (b, c, f, g) and centered(d, h).**

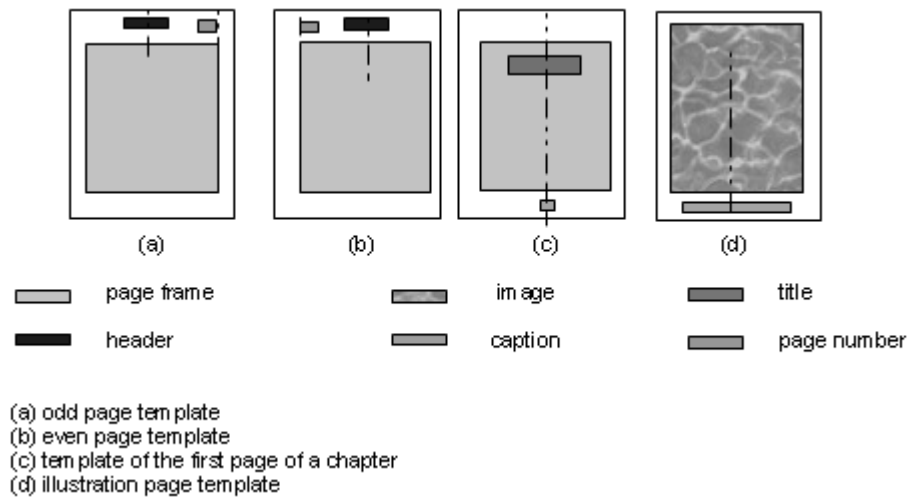
**Table 1: The geometrical relations used for inferring page templates.**

different geometrical relations L	justified	flush -left	flush-right	centered
<b>vertical axis</b>	(a)	(b)	(c)	(d)
<b>horizontal axis</b>	(e)	(f)	(g)	(h)

[6;7] uses a set of somehow similar relations: *left-to*, *right-to* and *aligned* for vertical relations and *above*, *aligned* and *below* for horizontal relations, while [12] uses the 13 Allen's relations. Our experiments show that the 4 chosen relations are relevant and informative enough to cover a very large part of page templates.

A relation  $R(a,b)$  between two elements has a value which characterizes more precisely the relation  $L$  between  $a$  and  $b$  (and is used when matching a page template against a page). The function used to compute this value depends upon  $L$ , and reflects an appropriate distance between  $a$  and  $b$ . Currently the vertical distance between  $a$  and  $b$  is used for relations (a), (b), (c), (d) and the horizontal distance between  $a$  and  $b$  for the relations (e), (f), (g), (h)

The template inference is possible at document level due to the repetition of pages sharing a same layout template. A minimal number of pages sharing the same template is then required by the method to produce reliable results. Before explaining in detail the method used for inferring the page templates, we will illustrate it with an example. Illustration 1 shows the page layout templates generated by our method from one document of our test collection (see Section Evaluation). The logical components used are these cited above.



**Figure 2: The different page templates generated from a document sample.**

Depending of the type of document, a single page template (as in a technical document) or several page templates (as in an illustrated book) can be used in one given document. In this example, two different page templates are used for odd and even pages (a frequent case for books). A specific page template is used for the first page of the chapters, and for illustrated pages (composed of the illustration and the caption). The formal template corresponding to the odd pages of the book (Figure 1(a)) is shown Table 2.

**Table 2: Example of geometrical relations describing a page.**

<pre>template#1= {{n#1,n#2,n#3}, {L1, L2, L3}}</pre>	<pre>L1(n#2,n#1) = {X-centered, 16}</pre>
<pre>n#1 = {header}</pre>	<pre>L2(n#2,n#3) = {X-right-ragged, 16}</pre>
<pre>n#2 = {pagebody}</pre>	<pre>L3(n#1,n#3) = {Y-justified, 98}</pre>
<pre>n#3 = {pagenumber}</pre>	

Interestingly but not surprisingly, a particular element plays a very specific role: the *page body*, the page zone where the foreground content is laid out. Page body (called *type area* by typographers as in [8]) is a key notion in page design as explained by Müller-Brockmann. The content is organized within this zone and elements share most of their relation with it. The use of page body to describe page template is one novelty in this work, and allows for the use of pure

typographical alignment relations between elements. This key layout element (page body) is often ignored by previous work, one exception is [10], which illustrates different advantageous uses of it.

### 3. THE METHOD

We will now describe the different steps that are performed for generating a set of page templates for a document from the 2008 INEX bookStructure competition, *Bible Myths*<sup>1</sup>. As input the method takes a document, which is represented as a set of pages, these pages containing textual and image elements defined as geometrical regions (common output after physical analysis). An example of such input is shown Figure 3.d.

#### Step 1: Run Logical Components

Available logical components are applied to the document in order to identify logical elements. In the rest of the paper, we call *labeled element*, a page element recognized by one of the components. Obviously the output of these components may be noisy, but we will see how the method is not only able to cope with noise, but also to correct some types of noise (see Section 2 for the list of components we use).

#### Step 2: Compute geometrical relations between labeled elements

For each page, all possible relations between labeled elements of this page are computed. The relations used are those given Table 1. With each page is associated the set of relations contained in it.

#### Step 3: Infer the list of page template candidates

All the sets of relations are collected and associated with the list of pages in which they occur. Table 2 shows some of the relations generated from the book shown as example and their frequency: the number of pages where they occur. The most frequent page template is composed of the following relations (PB: page body; PN: page number, header: page header), and corresponds to the even page template.

relation 1: XCentered(PB#1, header#0)

relation 2: Left(PB#1, PN#2)

relation 3: YJustified(PN#2, header#0)

The second page template corresponds to the odd page template. The third one is interesting since it corresponds to noisy pages where the page number was missed by the OCR engine. For these pages only the relation between page body and page header is extracted.

**Table 3: Set of geometrical relations generated. Read the first relation as: element PB in relation with header (XCentered) and PN (Left); and element PN in relation with header (YJustified)**

Page coverage (# pages)	Set of geometrical relations generated
210	XCentered(PB,header), Left(PB,PN), YJustified(PN,header)
201	XCentered(PB,header),Right(PB,PN), YJustified(PN,header)
22	XCentered(PB,header)
19	XCentered(PB,header),YJustified(PN,header)
18	Right(PB,PN)
16	Right(PB,PN), YJustified(PN,header)
12	XCentered(PB,header), Left(PB,header)

<sup>1</sup> available at [www.archive.org/details/biblemythsandthe00doanuoft](http://www.archive.org/details/biblemythsandthe00doanuoft)

	...
2	XCentered(PB,header), TOP(PB,header)
2	XCentered(PB,title), BOTTOM(PB,PN), Right(PN, title)
2	Xcentered(PB,header), Right(PB,PN), Left(PB,title)m YJustified(PN,header)

Then the value for each relation is computed (it specifies the distance between both related elements) by applying a function over the series of values provided by the labeled elements in the pages where the relation occurs. The function used is the arithmetic mean. In addition, if the standard deviation of the series is higher than a given threshold, the relation is not considered as a geometrical one (we suppose then that there is no fixed distance between both elements), and is discarded from the set of relations of the template. The values obtained for the most frequent template are (the value unit depends on the input document):

relation 1: XCentered(PB#1, header#0) = 17

relation 2: Left(PB#1, PN#2) = 16

relation 3: YJustified(PN#2, header#0) = 98

This means that the mean value between the elements PB and header for the first relation is 17 (pixels). We will refer to this page template as template #1 in the rest of the section.

#### Step 4: Page template assignment and correction

As a document may use several page templates we adopt a sequential covering strategy to infer them: among the list of template candidates, we first select the one which covers the greatest number of pages. We apply it over all pages using a fuzzy matching method to cope with noise. This allows for correcting cascaded errors such as labeling errors. The matched pages are discarded, and we select the next page template with the same method. The iteration stops when the remaining template candidates do not cover enough pages (a main parameter of the method).

##### Step 4.1: Select the page template candidate with the greatest page coverage

The set of relations (template candidate) with the largest list of pages is selected as best template candidate. In our example, this corresponds to template #1.

##### Step 4.2: Mark-up pages that partially match the current template (correction)

Once the page template is selected, it is matched against all pages of the document not already assigned to a page template. Since a page template can be viewed as a graph (each labeled element represents a node, and the relation between elements represents an edge), finding a match between a page template and a page can be considered as graph matching. [6] proposes for this problem a simplified version of the branch and bound search algorithm. Its solution allows for full or partial matching thanks to a two-step approach: first finding the best match, then trying finding some missing elements of the graph. Partial matching corresponds to a page where a set of elements of the template is properly recognized, but some are missing (not correctly recognized during Step 1 for instance). Once a partial matching is found, the missing relations allow for the determination of the location of the missing labeled elements. Unfortunately, [6] requires annotated data to setup weights in the used cost function. Since we do not want to leave the unsupervised paradigm, we implemented a fuzzy matching algorithm. This matching algorithm requires at least one correct labeled element in the current template (The algorithm will be explained in the long version, especially the use of the relation value discussed Step 3). Fuzzy matching is necessary in order to cope with noise. Noise can be categorized into the following categories:

1. **Missing labeled elements:** some labeled elements of the page template were not recognized during Step 1. If all elements of a page are missed, the current matching algorithm fails.
2. **Wrong labeled elements:** labeled elements of the template were incorrectly recognized.
3. **Missing text:** Objects were missed by OCR, and then cannot be labeled during Step 1.

From template #1, the following 55 pages are added thanks to fuzzy matching:

Added [55]: [8, 14, 22, 24, 26, 36, 78, 96, 192, 196, 210, 224, 278, 302, 312, 336, 372, 378, 380, 388, 422, 426, 430, 448, 460, 462, 464, 486, 492, 526, 528, 530, 534, 556, 568, 570, 574, 576, 578, 582, 584, 586, 588, 590, 596, 598, 600, 602, 604, 606, 608, 610, 612, 614, 616]

These pages were not in the initial list of pages because of labeling errors, mainly due to OCR errors which impacted page numbers and page header labeling.

Applying the sequential covering method over the page templates list, the final set of page templates generated from our book sample is the following (the minimal coverage of a template is set up to 5 pages):

**Table 4: List of final templates for the book *Bible Myths* (2008 Inex book 00)**

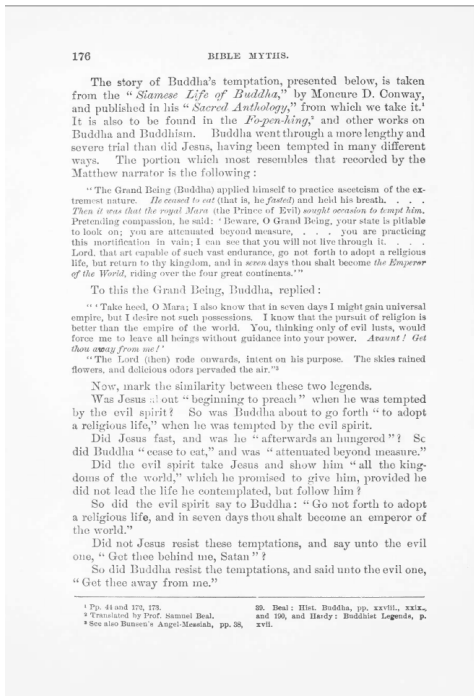
<p><b>TEMPLATE 1</b> (initially covered pages: 210, added by fuzzy match: 55)  header#0, PB#1, PN#2  XCentered[[ (PB#1, header#0)]] = 17  Left[[ (PB#1, PN#2)]] = 16  YCentered[[ (PN#2, header#0)]] = 98</p>	<p><b>TEMPLATE 3</b> (initially covered pages: 11, added by fuzzy match: None)  <b>(generated due to page number missed by OCR)</b>  header#12, PB#13  XCentered[[ (PB#13, header#12)]] = 18</p>
<p><b>TEMPLATE 2</b> (initially covered pages: 201, added by fuzzy match: 55)  header#6, PB#7, PN#8  XCentered[[ (PB#7, header#6)]] = 17  Right[[ (PB#7, PN#8)]] = 17  YCentered[[ (PN#8, header#6)]] = 66</p>	<p><b>TEMPLATE 4</b> (initially covered pages: 10, added by fuzzy match: 35)  Title#34, PB#35  XCentered[[ (PB#35, Title#34)]] = -390</p>

Templates #1, #2 and #4 correspond to expected templates (odd/even pages and first page of chapter). Template #3 is a noisy template which was generated due text missed by the OCR engine and corresponding to page numbers (Solution for fixing this specific type of zoning/OCR error has been developed, but is beyond the scope of this paper).

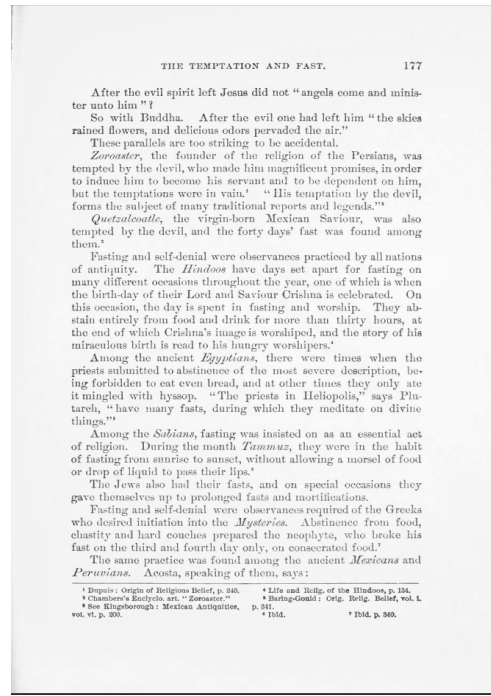
Finally there are 62 pages not covered by a page template in our book (bold numbers represent empty pages):

[1, 2, **3**, **4**, 5, 6, **10**, 12, 16, 18, 20, 25, **28**, 125, **138**, 189, 311, 379, 394, 445, 446, 456, 463, 544, 555, 559, **560**, 565, 567, 572, 580, 583, 585, 592, 594, **618**, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 640, **641**, **642**, 643, **644**]

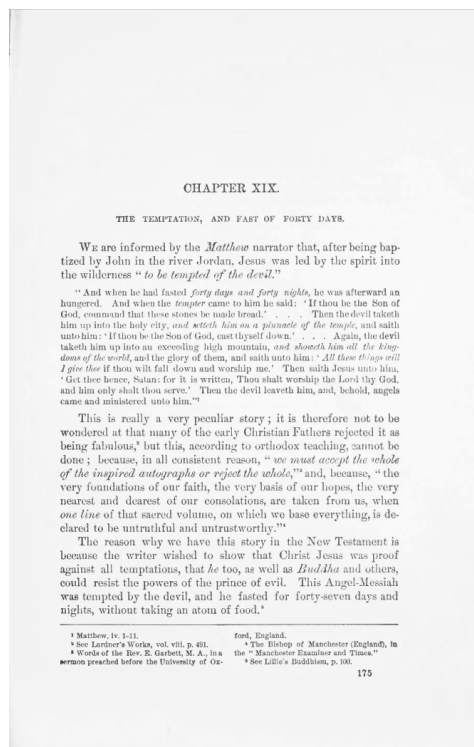
Most of them are part of the front or back-matter of the book, where no relation between labeled elements is found: the pages consist mainly in a page body (no page header or other significant element).



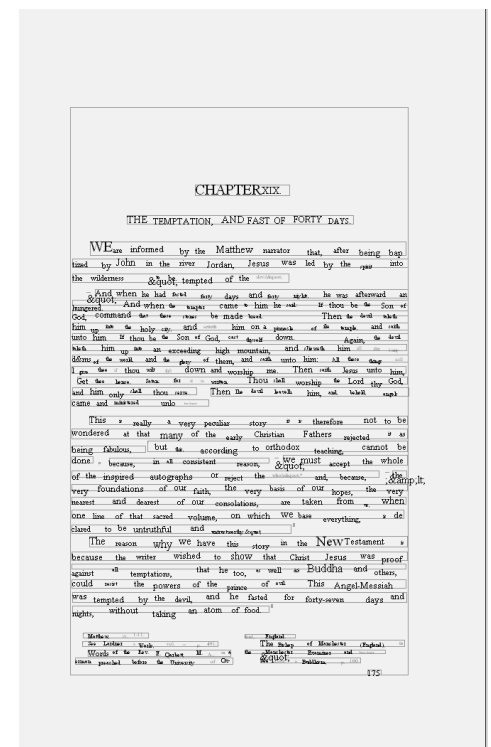
(a)



(b)



(c)



(d)

Figure 3: Example of pages corresponding to template #1 (a), template #2 (b), and template #3 (c). Figure (d) shows the segmentation corresponding to the page shown at (c).

## 4. EVALUATION

An indirect quantitative evaluation was performed which used a dataset from the INEX BookStructure task presented in [5]. The ground truth of the data corresponds to the hierarchical structure of 427 books, representing around 146,000 pages. The method is able to generate page templates for 411 books and fails to generate models for 16 books. During the second step, 27217 additional pages are associated with a template. Table 5 sums up these indicators. To be associated with a template, each labeled element of the template has to be identified in the page. The new associated pages correspond to pages with one or more errors, which were fixed by the matching algorithm (noisy non labeled elements are labeled thanks to the page template).

The 16 books without template correspond to relatively short books (altogether 1674 pages), where templates were composed of only one element: the page frame. Since this method only considers templates composed of at least 2 elements, no template is associated with these books. It is noteworthy that the page number detector completely fails to detect page numbers for these books, due to a too high level of noise for these elements (mostly ignored during the zoning step).

**Table 5: Coverage of templates before and after fuzzy matching**

<b>Total number of pages (427 books)</b>	146017
<b>Books without models (% of total pages)</b>	16 (1%)
<b>Number of pages associated with templates</b>	119412 (82%)
<b>Number of new pages associated with templates after fuzzy matching</b>	27217 (19% of the total pages)
<b>Number of white pages (no associated template)</b>	8674 (6%)

We do not manually annotate this collection to evaluate whether the page templates and their associated pages were correct. This required too much effort for the whole collection, and annotating a subset (how many books) is not convincing for us. Therefore we chose the option to indirectly evaluate our method. One key point of the method is the fuzzy matching step and we would like to assess whether the fuzzy matching step improves the result or simply introduces more errors. To achieve this, we focus on a specific page template that corresponds to the first page of chapters (see Figure 3.c). We are able to evaluate such a template thanks to the INEX corpus whose purpose is "test[ing] and compar[ing] automatic techniques for deriving structural information from digitized books in order to build a hyperlinked table of contents that could then be used to navigate inside the books" [5]. For this evaluation, we select pages associated with a page template containing an element labeled "Title". Such templates generally correspond to the first page of chapter, and must occur in the INEX ground-truth. We evaluate the set of the selected pages against the INEX ground-truth (Table 6). The initial step (no fuzzy match) provides a good precision (93.8%), but a low recall (55.6). After the fuzzy matching step, precision is still high (89.5, -4.1) and recall gains 10% up 65.6%. This evaluation might be considered as indicative of the effect of the fuzzy matching step for other templates: it should improve recall without penalizing too much precision.

**Table 6: Evaluation using the 2008 INEX BookStructure groundtruth. Only generated Template corresponding to chapter first pages are evaluated.**

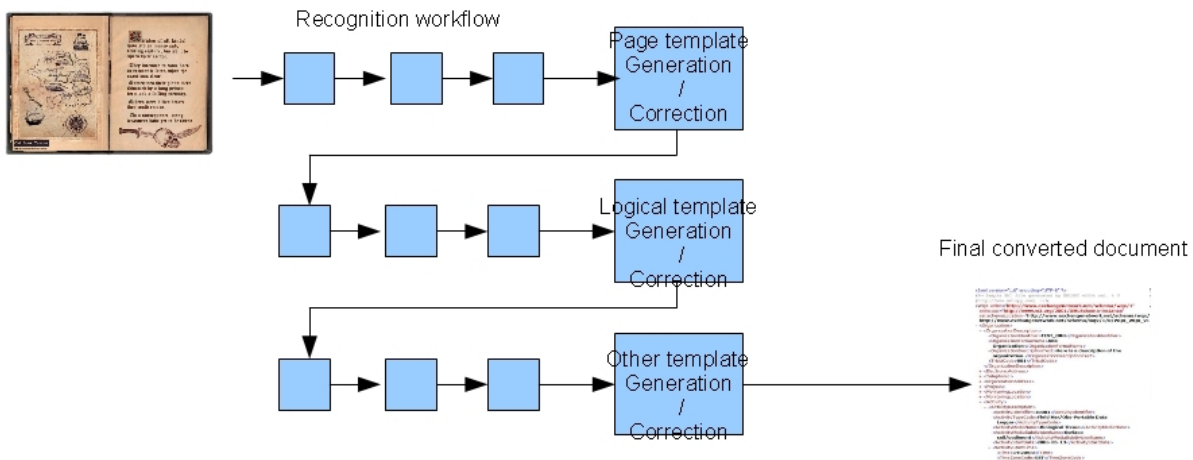
<b>"Title" templates without fuzzy matching</b>		<b>"Title" templates with correction</b>	
<b>Precision</b>	<b>Recall</b>	<b>Precision</b>	<b>Recall</b>
93.6	55.6	89.5 (-4.1)	65.1(+10)

Most of the errors captured by the correction step are due to labeling errors. In the case of the “title” templates, pages were initially not recognized by the template due to unlabeled titles missed by the ToC entries detector. We also found some errors due to software bugs, especially in the fuzzy matching algorithms which generate false positives. This is a general drawback of the methods: the more complicated models are, the more difficult applying them becomes. It is currently difficult to estimate the impact of such bugs before correcting them.

## 5. DISCUSSION

We have presented a first attempt to automatically generate page templates from a given document. A first prototype was implemented to validate the idea that page templates can be generated after logical analysis for generic documents. This prototype can be in many ways improved, but the first experiments are already encouraging: a very interesting feature of the method is its resilience to errors produced by previous steps and its ability to correct (to some extent) them: either zoning and OCR errors or tagging errors. A weakness of the method is its need for a set of logical labeling components, which is not always available to anyone. We are thinking of a solution which will relax these constraints by replacing these logical components by more general ones. Another extension is the introduction of page flow: page templates are not randomly chosen and each page template corresponds to a specific use (such as beginning of chapter, back and front matters). This will generate a document template composed of a constraint set of page templates. Although we thought this quite straightforward, this modeling requires more effort.

This work focuses on page template, but other kinds of templates can be considered targeting a specific organization of a document: content (text or image), and logical structure. An interesting usage of generating different types of templates is its insertion in a document analysis system (Figure 4). In a given conversion workflow, steps corresponding to template generation can be added in several points depending on the type of template in order to correct errors and also to control the current quality of the workflow. Generally these checkpoints could be used in order to provide feedback to the system regarding processing quality. These hypotheses are still exploratory but the first tests are promising and could contribute to the general reflection on document analysis system design.



**Figure 4: Template generation steps can be added to a workflow to correct and control partial outputs.**

## REFERENCES

1. Thomas Breuel, tutorial DAS, <http://ocrocourse.iupr.com/layout-analysis>
2. Michelangelo Ceci, Margherita Berardi, and Donato Malerba, *Relational Data Mining and ILP for Document Image Understanding*. In *Applied Artificial Intelligence*, 21:317-342, 2007.

3. Glen Ford, George R. Thoma, Mars project: Ground truth data for document image analysis, <http://archive.nlm.nih.gov/pubs/ford/groundTruthData/gtd.php>, National Library of Medicine, 2003.
4. Robert M. Haralick, Document Image Understanding: Geometric and Logical Layout, (section 2), CVPR94
5. INEX Book Structure Extraction Track, <http://users.info.unicaen.fr/~doucet/StructureExtraction2009/>
6. Jian Liang and David Doermann, M. Ma and J. K. Guo, "Page classification through logical labeling", *International Conference on Pattern Recognition 2002*, pages 477-480, Quebec, Canada, 2002.
7. Jian Liang and David Doermann, "Logical labeling of document images using layout graph matching with adaptive learning", *The Fifth International Workshop on Document Analysis Systems*, pages 224-235, Princeton, NJ, 2002.
8. Josef Müller-Brockamn, Grid System in graphic design, Niggli, 2007.
9. Faisal Shafait, Geometrical Layout Analysis of Scanned Documents, PhD dissertation, TU Kaiserslautern, Germany, 2008.
10. Faisal Shafait and Joost Van Beusekom and Daniel Keysers and Thomas M. Breuel, Structural Mixtures for Statistical Layout Analysis, IUPR 2008.
11. Christian K. Shin and David Doermann, Classification of document page images based on visual similarity of layout structures. Proceedings of the SPIE Document Recognition and Retrieval VII, 2000.
12. Hanno Walischewski, *Automatic Knowledge Acquisition for spatial Document Interpretation*, In proceedings of the Fourth International Conference Document Analysis and Recognition (ICDAR'97) 1997.