Towards a Canonical and Structured Representation of PDF Documents through Reverse Engineering.

Maurizio Rigamonti, Jean-Luc Bloechle, Karim Hadjar, Denis Lalanne and Rolf Ingold
DIVA group, University of Fribourg, Ch. Du Musée 3, CH-1700 Fribourg, Switzerland
{firstname. lastname}@unifr.ch

Abstract

This article presents Xed, a reverse engineering tool for PDF documents, which extracts the original document layout structure. Xed mixes electronic extraction methods with state-of-the-art document analysis techniques and outputs the layout structure in a hierarchical canonical form, i.e. which is universal and independent of the document type. This article first reviews the major traps and tricks of the PDF format. It then introduces the architecture of Xed along with its main modules, and, in particular, the document physical structure extraction algorithm. Later on, a canonical format is proposed and discussed with an example. Finally the results of a practical evaluation are presented, followed by an outline of future works on the logical structure extraction.

1. Introduction

Even though PDF (Portable Document Format [1]) became the universal Internet document exchange format, actual PDF writers are only targeting the layout preservation and do not reveal the actual physical and logical structures, which could drastically improve search and retrieval, due to a more precise indexing.

The PDF format has improved over the years and it offers rich syntax and numerous functionalities. Unfortunately, these improvements have increased PDF's complexity and further its functionalities have become difficult to use. In a previous work [6] we have already presented Xed (eXtracting Electronic Documents), a tool for extracting the hidden layout structures of PDF documents. The objectives of this paper are 1) to recover a PDF document layout structure, in order to improve the pertinence of the searches, 2) to *purify* it, eliminating inconsistencies and superfluous information, and finally 3) to store this cleaned layout structure in a canonical form, i.e. reduced to the simplest and most significant form possible without loss of generality.

This lack of inner layout structures means that PDF efficiency inside the web-based search-engines for information retrieval can be improved, but also that end users would not be able to copy-paste textual parts of a document from PDF viewers maintaining the reading order. A reconstruction of the homogeneous text entities (words, lines, and blocks) extracted from a PDF file is required before any use of the textual content. A partial evaluation of state-of-the tools and existing researches [2,3,4,5,7,8] on these issues has already been presented in [6].

The previous version of Xed transformed PDF files into SVG or semi-structured XML format. In this paper we propose a canonical format for storing both the newly extracted information, i.e. the various document structures, and the purified PDF file. The latter representation allows a document-independent description of physical structures.

This paper is organized as follows: in section 2 we present a subset of PDF abnormalities. Section 3 describes Xed architecture. The following one presents the canonical format explanation and the physical layout restitution algorithm. The section 5 resumes an evaluation of the current prototype and, finally, the last section concludes this paper and gives some perspectives for future works.

2. Inspecting PDF Files

This section present some abnormalities and traps discovered while extracting PDF documents and legitimate the *need* for building homogenous text entities, which will benefit of the use of image processing techniques. These traps belong to three main categories: segmentation errors, font representation abnormalities and document re-editing.

The most current trap consists of over-segmented words. This problem depends on PDF writers, which decompose words when typographic properties change (in particular the kerning, i.e. the offset between two characters.). PDF format offers instructions conceived to represent kerning modification in a string, but very often PDF documents contain separated sub-strings, preserving visualization instead of content.

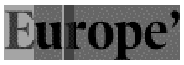


Figure 1. The over-segmented word "Europe'".

Figure 1 illustrates a singular over-segmentation error, extracted from *International Herald Tribune* newspaper (from February 21, 2005). The word "Europe'" is split in three sub-strings, and a blank space has been added between sub-strings "u" and "rope'" (in Figure 1, the dark gray box superposed to 'r' character is the addition of " " and of "rope'" bounding boxes). This artifact has been revealed in different editions of *International Herald Tribune*.

Fonts re-mapping is another frequent trap. PDF allows encapsulating proprietary fonts in the document file. Those ones are represented with font files that contain typographic information and character descriptions. Usually, the latter are defined by *codes* (e.g. 97), *glyphs* (e.g. "Asmall") and *charstrings* (the shape of the printed character). Some documents redefine the codes of a well-defined encoding: for example, the code 97 (usually 'a') is assigned to character 'r'. The consequence is that the visualization is correct, but not the extracted content. The French newspaper *Le Monde* abuses of this technique for advertising and for its own name.

Another current trick is the font splitting. While defining a font file, there is the possibility to encapsulate only sub-dictionaries of characters used in the document (i.e. part of the font). Actually, some documents (e.g. *Le Monde*) use this capability to split a font file in more sub-dictionaries with different names. So, it's obvious that the font name cannot be considered as a criterion to detect homogenous text entities.

A different category of traps consists of unmeaning information added by humans. This one is typical in case of document layout correction using WYSIWYG editors. For example, the user aligns paragraphs using blank characters, instead of defining paragraph properties. It means that the PDF contains information invisible at print, but modifying the mean of electronic content.

The list of examples presented in this section is not exhaustive, but introduces and justifies the extraction of a canonical format, aiming at cleaning and restructuring PDF documents. The following section presents Xed and in particular the various steps for purifying and structuring a PDF document.

3. Structure of Xed

Xed is a tool developed in Java for reverse engineering PDF documents, which targets to purify and to enrich PDF documents with their original structures, using the most common paradigm of document image analysis: extraction of raw primitives, filtering, clustering of homogeneous regions and interpretation.

Figure 2 presents Xed architecture and the different phases of extraction and analysis.

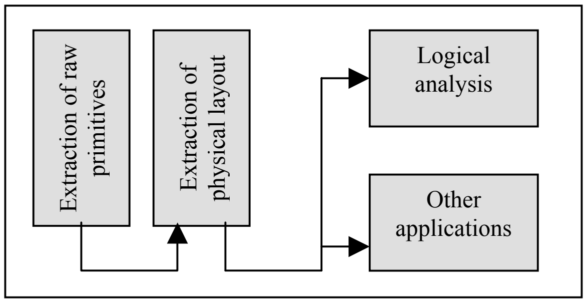


Figure 2. Architecture of Xed.

The first phase consists of extracting the raw primitives: graphics, text and images. The process has been presented with more details in [6], and can be summarized as follow:

1. Parsing of the PDF File and construction of a tree containing normalized PDF objects (e.g. using a unique representation for each type);
2. Interpretation of the tree in order to create the *virtual document*, i.e. an internal and application-specific printing of the PDF document. The virtual document is represented such as a tree, which is important for followings phases;
3. Save of the tree in a SVG file for each document page, allowing users to evaluate the qualitative extraction of the original PDF document.

The extracted primitives are first classified per category (text, image or graphics), without introducing ontologies, supplementary manipulations (except fonts re-mapping) or analysis. At this stage, it is impossible to ensure that a word is classified as text: in some cases, a word is represented with segments and thus considered as a graphic primitive. The validation of primitives can be accomplished *a posteriori* using image analysis methods and symbolic interpretation, but currently Xed does not provide those techniques.

The second phase is the physical structure extraction, which aims at recognizing homogenous entities, and in particular textual chunks, such as blocks, lines and words. The document properties concerned in this phase are typographic (i.e. font sizes), topologic (the distance between text entities) and syntactic (the string type). The physical layout extraction modifies the tree constructed in the previous phase and ends with the reorganization of the document in the *canonical format*. This phase is explained in details in the next section.

The last phase aims at extracting the logical structures of documents, which describes the mean of text entities and the hierarchical relationships between them. In the case of Xed, the main steps involved in the logical structure extraction are:

1. Textual entities labeling. For example, the various entities of a newspaper article can be labeled as title, author, body, etc;
2. Document modeling: labeled entities are projected in a document model. For each class of document (e.g. *Le monde* newspaper front page), a model defines its grammar, i.e. a hierarchy of logical entities. For example, a newspaper is composed of articles, which is made of a title, an author, a body, which is composed of paragraphs, etc.

It is important to notice that a textual block in the physical layout does *not* always correspond to a logical one (e.g. a textual block can contain two paragraphs, or an article can contain several physical textual blocks.). Currently, a logical analyzer, called DOLORES, is in development, but other logical analyzers (or applications) can extend the previous phases of Xed, in respect of various users needs and of different classes of documents.

4. The canonical format

This section presents the canonical format for representing electronic documents in a standard way and the document layout structure extraction algorithm.

4.1. Concept and foundations

Initially, Xed's reorganization of PDF documents consisted in the restitution of textual entities and reading order[1] in SVG (or in an XML format derived from the first one). The SVG format allowed to encapsulate a cluster of homogenous entities in a group and to create a hierarchy with the latter. Unfortunately, its use highlighted that reading an SVG file was quite difficult for end-users, and that the hierarchy of SVG document did not clarify the function of textual entities and groups (i.e., page, block, etc.) The need for representing information in a canonical standardized way became an unavoidable requirement.

Currently, Xed results are expressed in a canonical format, based on the following foundations:

1. All the primitives contained in the original document (text, graphics and images) can be represented in a *non-ambiguous* manner;
2. The resultant document is *unique* in spite of PDF writer;
3. Further research and application based on Xed profits of a handily medium, which allows overcoming the way information is represented, encoded or organized;
4. The format can be easily used to represent results of document analysis methods too.

Intuitively, the main task achieved by this format is to structure a document into a set of well-defined primitives, where the textual content of a page is segmented into blocks, which are themselves divided into lines and, finally, the latter into consecutives words, punctuations, number, signs and blanks. Graphics primitives are labelled as horizontal threads, vertical threads, frames or general paths. Finally, the images are represented with information on the area textured in the original document and with the reference to original source file. The following partial DTD emphasises document hierarchy and describes the main primitives of the canonical format in more details:

```
<!ELEMENT document (fontlist?, page+)>
<!ELEMENT fontlist (font+)>
<!ELEMENT page (image*, graphic, htext*,
               vtext*)>
<!ELEMENT graphic (hthread*, vthread*, frame*,
                  gtext*, path*)>
<!ELEMENT path (gline,cubic,quadratic)*>
<!ELEMENT htext (line*)>
<!ELEMENT vtext (line*)>
<!ELEMENT line (chars*)>
<!ELEMENT chars (#PCDATA)>
```

4.2. Layout structure extraction algorithm

This section presents the algorithm for extracting and representing the physical layout in a canonical format and the method is illustrated using three meaningful cases of newspapers analysis. In fact, the selected dataset, on which the algorithm has been developed and refined, consists in documents with complex layout (i.e. 30 newspapers). No evaluation has been yet performed on documents with simpler structures (papers, technical reports, etc.). Figure 3 shows the primitives extraction results for newspapers.

[1] Restitution of reading order is a difficult problem, in particular when dealing with newspapers, where also psychological concepts are implied. We consider simply that reading order is respected in homogenous textual entities such as blocks and lines.

These examples show clearly that it is impossible to make any assumption about the textual segmentation of the raw primitives extracted from PDF files.

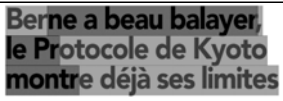
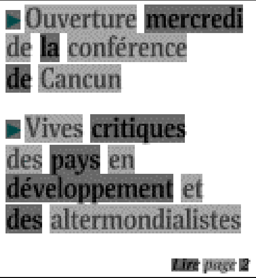
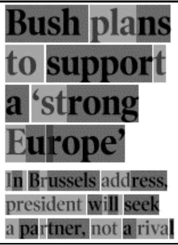
La Liberté	 Berne a beau balayer, le Protocole de Kyoto montre déjà ses limites
Le Monde	 Ouverture mercredi de la conférence de Cancun Vives critiques des pays en développement et des altermondialistes Lire page 2
International Herald Tribune	 Bush plans to support a 'strong Europe' In Brussels address, president will seek a partner, not a rival

Figure 3. Different segmentations examples.

In general, *La Liberté* is segmented in lines, but each time a kerning appears the lines are split; On the other hand, *Le Monde* is segmented into words; finally, the segmentation of the *International Herald Tribune* seems completely chaotic.

The algorithm transforms the tree of primitives (the virtual document presented in previous section) in a different tree, which represents the document in the canonical format. Currently, the algorithm is composed of eight main steps:

1. Trim all text primitives in order to remove all superfluous blanks (part of text blocks as well as standalone). This step is important because sometimes a word could have white spaces between its own letters (cf. the string "Europe'" in Figure 1, section 2);
2. Merge trimmed text primitives horizontally to obtain new strings, given an empirical distance threshold;
3. Tokenise the strings using a separators list and obtain isolated words, meaningful blanks, punctuations and special characters;
4. Merge horizontally the strings into lines with an empirical distance threshold;
5. Add required nonexistent blanks between two consecutive strings of a line;
6. Merge the lines vertically into blocks given an empirical distance threshold (in this case the average distance between lines);
7. Retroactive merge. Reanalyse blocks content in order to recompose over-segmented lines. This over-segmentation is inducted from the low thresholds selected in step 4: in case of justified lines (Figure 4) the distance between strings augments and the lines cannot be merged. The main criterion for retroactive merging is that horizontal distance is always constant between strings on the same line;

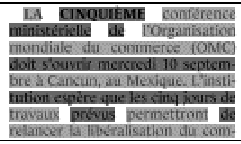
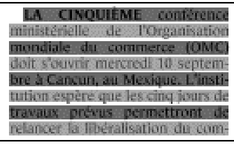
 LA CINQUIÈME conférence ministérielle de l'Organisation mondiale du commerce (OMC) doit s'ouvrir mercredi 10 septem-bre à Cancun, au Mexique. L'insti-tution espère que les cinq jours de travaux prévus permettront de relancer la libéralisation du com-	 LA CINQUIÈME conférence ministérielle de l'Organisation mondiale du commerce (OMC) doit s'ouvrir mercredi 10 septem-bre à Cancun, au Mexique. L'insti-tution espère que les cinq jours de travaux prévus permettront de relancer la libéralisation du com-

Figure 4. Result of retroactive merging.

8. Parse again all strings to merge consecutive blanks or dots and tag strings with the syntactical attributes "word", "number", "blank", "sign" and "punctuation".

The eight steps of the algorithm presented here have been simplified and adapted for western newspapers; some experiments with Asiatic newspapers has shown interesting results, but we were unable to evaluate them precisely with our poor knowledge of the language. Actually, we sort strings, lines and blocks before each merging in order to maintain the reading order. Selected thresholds are based on font size, defined from empirical experiences with different classes of documents, and are always minimal, over-segmentation being preferable than the opposite.

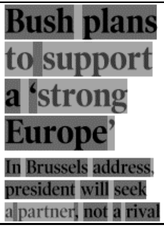
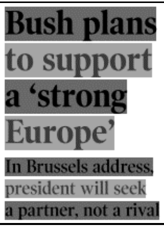
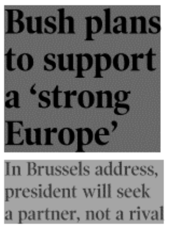
 Bush plans to support a 'strong Europe' In Brussels address, president will seek a partner, not a rival	 Bush plans to support a 'strong Europe' In Brussels address, president will seek a partner, not a rival
 Bush plans to support a 'strong Europe' In Brussels address, president will seek a partner, not a rival	Bush plans to support a 'strong Europe' In Brussels address, president will seek a partner, not a rival

Figure 5. Algorithm results.

Figure 5 illustrates the algorithm results using the *International Herald Tribune* example (see Figure 3), the syntactical segmentation, lines and blocks merged and, finally, the plain text extracted. Finally, Figure 6 is an overview of the canonical XML file generated for this example (for concision purpose, only the tags are showed, the attributes being hidden on this graphical representation).

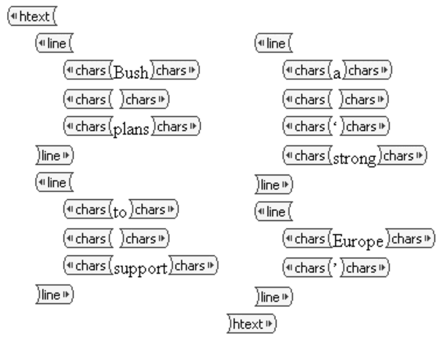


Figure 6. Overview of canonical XML file

5. Experimental results

The evaluation of the physical layout extraction in a canonical format has been performed on a set of representative newspapers front pages, presenting different types of segmentation of textual primitives: *La Liberté*, *Le Monde* and the *International Herald Tribune*. The canonical layout structure of 30 documents, i.e. ten front pages per newspaper type, have been extracted and evaluated in respect to human judgments. The percent of words, lines and blocks correctly segmented are presented in table 1.

Table 1: Evaluation.

	La Liberté	*Le Monde*	*International Herald tribune*
% of correct words	99.90	99.94	99.94
% of correct lines	99.24	99.57	99.47
% of correct blocks	97.00	98.26	98.96

6. Conclusion

The extraction of physical and logical structures from PDF documents is a difficult problem with many potential practical applications. The reverse engineering of PDF is confronted with multiple traps and requires the development of low-level tools for extracting the PDF primitives (text, graphic and images) and lost document structures (physical layout, reading order, logical structures, etc.). A canonical format has been proposed in this article as a standard for representing purified electronic documents and for enriching documents with theirs structures (and others derived data). Xed is the tool we developed in order to accomplish this task. Its originality consists in merging 1) primitives extraction from PDF with 2) classical document image techniques. The preliminary evaluation encourages us to refine the minor problems in the primitive extraction and in the layout analysis and to undertake new research topics. In particular, our future work will deal with the logical structures extraction based on the canonical layout structure and with the study of the structural relationships between different classes of documents. The final scope of our work is the purification and the regeneration of PDF documents along with their physical and logical structures.

7. References

[1] Adobe, "Adobe PDF Reference",
http://partners.adobe.com/asn/tech/pdf/specifications.jsp

[2] A. Anjewierden, "AIDAS: Incremental logical structure discovery in PDF document", ICDAR'01, Seattle (USA), September 2001, pp. 374-377.

[3] A. Anjewierden and S. Kabel, "Automatic indexing of documents with ontologies", In 13th Belgian/Dutch Conference on Artificial Intelligence (BNAIC), Amsterdam (Holland), October 2001, pp. 23-30.

[4] S. R. Bagley, D. F. Brailsford and M. R. B. Hardy, "Creating reusable well-structured PDF as a sequence of component object graphic (COG) elements", Proceeding of the 2003 ACM symposium on Document engineering, Grenoble (France), November 2003, pp. 58-67.

[5] H. Chao and J. Fan, "Layout and Content Extraction for PDF Documents", DAS'04, Florence (Italy), September 2004, pp. 213-224.

[6] K. Hadjar, M. Rigamonti, D. Lalanne and R. Ingold, "Xed: a new tool for eXtracting hidden structures from Electronic Documents", DIAL 2004, Palo Alto (USA), January 2004, pp. 212-221

[7] W. S. Lovegrove and D. F. Brailsford, "Document analysis of PDF files: methods, results and implications", *Electronic publishing*, September 1995, pp. 207-220.

[8] M.D. Paknad and R.M. Ayers, "Method and apparatus for identifying words described in a portable electronic document", U.S. Patent 5,832,530, November 1998