Daniel@FinTOC-2021: Taking Advantage of Images and Vectorial Shapes in Native PDF Document Analysis

Emmanuel Giguet
Normandie Univ, UNICAEN,
ENSICAEN, CNRS, GREYC
14000 Caen, France
emmanuel.giguet@unicaen.fr

Gaël Lejeune
STIH, Sorbonne University
75005 Paris, France
gael.lejeune@sorbonne-universite.fr

Abstract

In this paper, we present our contribution to the FinTOC-2021 Shared Task "Financial Document Structure Extraction". We participated in the tracks dedicated to English and French document processing. We get results for Title detection and TOC generation performance which demonstrates a good precision. We address the problem in a fairly unusual but ambitious way which consists in considering simultaneously text content, vectorial shapes and images embedded in the native PDF document, and to structure the document in its entirety.

1 Introduction

In the Fintoc-2021 Financial Document Structure Extraction competition (El Maarouf et al., 2021), two tasks have been proposed by the organizers : *ToC Structure Extraction* and *Title Detection*.

Extracting a table of content (ToC) is a way to improve information access in large documents, it can also be a mean to improve the results of some Natural Language Processing or Document Analysis tasks, for instance document classification and clustering (Doucet and Lehtonen, 2007; Ait Elhadj et al., 2012), converting a document in a format in which the structure is important (Marinai et al., 2010) or navigating through electronic documents(Déjean and Meunier, 2005). A bigger motivation lies in the fact that having a proper Table of Contents is a way to handle documents like a structured set of paragraphs rather than a simple sequence of strings or sentences where the logical structure disappeared. This logical structure often happens with generic OCR tools whose output is simply a character string where the only structural aspect that has been kept is the pages and other structural properties have disappeared during the automatic processing (Lecluze and Lejeune, 2014). Two different subtasks lie behind ToC extraction : (i) extracting a logical structure that has been explicitly marked in a ToC and map it to the titles present in the document, and (ii) creating a ToC when no one is present in the document by spotting the titles and their hierarchy. In both cases, systems need to be able to perform Title Detection in order to get candidates to populate the ToC. Detecting titles has also an interest on its own since it has also been used as a feature for different tasks like text classification (Lejeune et al., 2013), Terminology Acquisition (Daille et al., 2016) or Keyphrase Extraction (Florescu and Caragea, 2017). The paper is organized as follows. In Section 2 we present the datasets. In Section 3, we describe our method for both tasks. the method we designed. In Section 4 we present a discussion about our results, and we draw some perspectives for future work.

2 Data

The training set and test set of the shared tasks are composed of financial prospectuses written in French and English. The documents are distributed as native PDF documents.

The structure of the prospectuses is not standardized, which helps to have a real-world case where there is variation in the data. However, we expect the presence of some particular sections and an overall structure which will not vary too much. The format and layout varies greatly from one document to another and is often complex, with tables, nested lists of numbered and bulleted items, framed content, graphs, columns. The majority of prospectuses are published without a table of content (ToC), which means you can not rely on a ToC detection and parsing module to achieve the tasks. All this makes the challenge interesting. A comprehensive description of the datasets can be found in (El Maarouf et al., 2021).

3 Method

The experiment is conducted on native PDF documents. In line with the work presented in FinSBD-2 task by (Giguet and Lejeune, 2021), we choose to implement an end-to-end pipeline from the PDF file itself to a fully structured document. This approach allows to control the entire process. Titles and Table of Contents that we generate for the shared tasks are derivative outputs of the system.

3.1 Document Preprocessing

The document content is extracted using the `pdf2xml` command (Déjean, 2007). Three useful types of content are extracted from the document: text, vectorial shapes, and images.

Text Preprocessing

`Pdf2xml` introduces the concepts of token, line and block, as three computational text units. We choose to only rely on the "token" unit. In practice, most output tokens correspond to words or numbers but they can also correspond to a concatenation of several interpretable units or to a breakdown of an interpretable unit, depending on character spacing. We choose to redefine our own "line" unit in order to better control the coherence of our hierarchy of graphical units. We abandon the concept of "block" whose empirical foundations are too weak.

Vectorial Shapes Preprocessing

Using `pdf2xml` allows to rely on vectorial information during document analysis. Text background, framed content, underline text, table grid are crucial information that contributes to sense making. They simplify the reader's task, and contribute in a positive way to automatic document analysis.

Most vectorial shapes are basic closed path, mostly rectangles. Graphical lines or graphical points do not exist: lines as well as points are rectangles interpreted by the cognitive skills of the reader as lines or points. In order to use vectorial information in document analysis, we implemented a preprocessing stage that builds composite vectorial shapes and interprets them as background colors or borders. This preprocessing component returns shapes that are used by our system to detect framed content, table grids, and text background. It improves the detection of titles which are presented as framed text and it avoids considering table headers as titles.

Images Preprocessing

`Pdf2xml` extracts images from the pdf. They may be used in different context such as logos in the title page, figures in the document body. An other interesting feature lies in the fact that certain character symbols are serialized as images, in particular specific item bullets such as arrows or checkboxes. They are indistinguishable from a standard symbol character by the human eye.

We choose to handle images as traditional symbol characters, so that they can be exploited by the structuration process, in particular by the list identification module. Identical images are grouped, and a virtual token containing a fake character glyph is created. The bounding box attributes are associated to the token and a fake font name is set. These virtual tokens are inserted at the right location by the line builder module thanks to the character x-y coordinates. This technique significantly improves the detection of list items and, as a consequence, the recognition of the global document structure.

3.2 Document Structure Parsing

Page Layout Analysis

Page Layout Analysis (PLA) aims at recognizing and labeling content areas in a page, e.g., text regions, tables, figures, lists, headers, footers. It is the subject of abundant research and articles (Antonacopoulos et al., 2009).

While PLA is often achieved at page scope and aims at bounding content regions, we have taken a model-driven approach at document scope. We try to directly infer Page Layout Models from the whole document and we then try to instantiate them on pages.

Our Page Layout Model (PLM) is hierarchical and contains 2 positions at top-level: the *margin area* and the *main content area*. The *margin area* contains two particular position, the *header area* located at the top, and the *footer area* located at the bottom. *Aside areas* may contain particular data such as vertically-oriented text. The *main content area* contains *column areas* containing text, figures or tables. *Floating areas* are defined to receive content external to column area, such as large figures, tables or framed texts.

The positions that we try to fill at document scope are header, footer and main columns. First, pages are grouped depending on their size and orientation (i.e., portrait or landscape). Then header area and footer area are detected. Column areas are in the model but due to time constraints, the detection module is not fully implemented in this prototype yet.

Detecting Header and Footer Areas

Header and footer area boundaries are computed from the repetition of similar tokens located at similar positions at the top and at the bottom of contiguous pages (Déjean and Meunier, 2006). We take into account possible odd and even page layouts. The detection is done on the first twenty pages of the document. While this number is arbitrary, we consider it is enough to make reliable decisions in case of odd and even layouts.

A special process detects page numbering and computes the shift between the PDF page numbering and the document page numbering. Page numbering is computed from the repetition of tokens containing decimals and located at similar positions at the top or at the bottom of contiguous pages. These tokens are taken into account when computing header and footer boundaries.

Detecting the Table of Contents

The TOC is located in the first pages of the document. It can spread over a limited number of contiguous pages. One formal property is common to all TOCs: the page numbers are right-aligned and form an increasing sequence of integers.

These characteristics are fully exploited in the core of our TOC identification process: we consider the pages of the first third of the document as a search space. Then, we select the first right-aligned sequence of lines ending by an integer and that may spread over contiguous pages.

Linking TOC Entries and Headers

Linking Table of Content Entries to main content is one of the most important process when structuring a document (Déjean and Meunier, 2010). Computing successfully such relations demonstrates the reliability of header detection and permits to set hyperlinks from toc entries to document headers.

Once TOC is detected, each TOC Entry is linked to its corresponding page number in the document. This page number is converted to the PDF page number thanks to the page shift (see section 3.2). Then header is searched in the related PDF page. When found, the corresponding line is categorized as header.

Table Detection

Table detection to exclude table content from the main text stream. It allows to exclude tables when searching for list items, sentences or titles.

The table detection module analyzes the PDF vectorial shapes. Our algorithm builds table grids from adjacent framed table cells. The framed table cells are built from vectorial shapes that may represent cell borders. The table grid is defined by the graph of adjacent framed table cells.

Unordered List Structure Induction

Unordered lists are also called *bulleted lists* since the list items are supposed to be marked with bullets. Unordered lists may spread over multiple pages.

Unordered list items are searched at page scope. The typographical symbols (glyphs) used to introduce items are not predefined. We infer the symbol by identifying multiple left-aligned lines introduced by the same single-character token. In this way, the algorithm captures various bullet symbols such as squares, white bullets... Alphabetical or decimal characters are rejected as possible bullet style type. Images of character symbols are transparently handled thanks to virtual tokens created during the preprocessing stage.

The aim of the algorithm is to identify PDF lines which corresponds to new bulleted list item (i.e., list item leading lines). The objective is not to bound list items which cover multiple lines. Indeed, the end of list items are computed while computing paragraph structures: a list item ends when the next list item starts (i.e., same bullet symbol, same indentation) or when less indented text objects starts.

Ordered List Structure Induction in PDF Documents

Ordered list items are searched at document scope. We first select numbered lines thanks to a set of regular expressions, and we analyse each numbering prefix as a tuple $\langle P, S, I, C \rangle$ where *P* refers to the numbering pattern (string), *S* refers to the numbering style type (single character), *I* refers to the numbering count written in numbering style type (single character), and *C* refers to the decimal value of the numbering count (integer).

The numbering style types are defined as follows: Decimal (D), Lower-Latin (L), Upper-Latin (M), Lower-Greek (G) Upper-Greek (H), Lower-Roman (R), Upper-Roman (S), Lower-Latin OR Lower-Roman (?), Upper-Latin OR Upper-Roman (!).

To illustrate, the line “A.2.c) My Header" is analysed as ⟨ A.2.L), L, c, 3 ⟩.

Lines are grouped in clusters sharing the same numbering pattern. A disambiguation process assigns an unambiguous style type to ambiguous lines. The underlying strategy is to complement unambiguous yet incomplete series in order to build coherent, ordered series.

Paragraph Structure Induction

The aim of paragraph structure induction is to infer paragraph models that are later used to detect paragraph instances. The underlying idea to automatically infer the settings of paragraph styles.

Paragraphs are complex objects: a canonical paragraph is made of a leading line, multiple body lines and a trailing line. The leading line can have positive or negative indentation. In context, paragraphs may be visually separated from other objects thanks to above spacing and below spacing.

In order to build paragraph models, we first identify reliable paragraph bodies: sequences of three or more lines with same line spacing and compatible left and right coordinates. Then, leading lines and trailing lines are identified considering same line spacing, compatible left and/or right coordinates (to detect left and right alignments), same style. Paragraph lines are categorized as follows: L for leading line, B for body lines, T for trailing line. Header lines are categorized H. Other lines are categorized as ? for undefined.

In order to fill paragraph models, paragraph settings are derived from the reliable paragraphs that are detected. When derived, leading lines of unordered and ordered list items are considered to create list item models.

Once paragraph models and list item models are built, the models are used to detect less reliable paragraphs and list items (i.e., containing less than three body lines). Compatible models are applied and lines are categorized L, B (if exists) or T (if exists). Remaining undefined lines are categorized considering line-spacing.

4 Results and discussion

The document-wise approach we presented was evaluated on both tasks of FinTOC 2021 : *Title Detection* and *Table of Content extraction*.

In table 1 and 2 we present the results we obtained respetively on the *Title Detection* and the *ToC Extraction* tasks. The results we obtain shows an overall good precision on both languages but a quite low recall. These results are encouraging when we consider all the structures that we want to handle, though our system gives too much False Negatives for title detection and consequently for ToC extraction.

Table 1: Results for Title Detection

	Prec	Rec	F1
fr	0.842	0.485	0.606
en	0.913	0.338	0.465

Table 2: Results for ToC Extraction

	Prec	Rec	F1
fr	49.7	28.6	35.8
en	52.8	18.6	25.1

The rationale of our method is to have an end-to-end pipeline from the PDF file itself to a fully structured document, it seems that this is a good way to avoid false positives. The steps comprised in our method (layout analysis, header/footer detection, list detection and paragraph induction) seem to act as filters to avoid an over structuration of the document. The consequence is that the results we obtain are very encouraging in terms of precision, but the recall remains quite low. Still, we believe there is a great interest in representing a fairly unusual but ambitious way to deal with the document structure as a whole.

References

Ali Ait Elhadj, Mohand Boughanem, Mohamed Mezghiche, and Fatiha Souam. 2012. Using structural similarity for clustering XML documents. *Knowledge and Information Systems*, 32(1):109–139.

Apostolos Antonacopoulos, David Bridson, Christos Papadopoulos, and Stefan Pletschacher. 2009. A realistic dataset for performance evaluation of document layout analysis. In *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, pages 296–300.

Béatrice Daille, Evelyne Jacquey, Gaël Lejeune, Luis Felipe Melo, and Yannick Toussaint. 2016. Ambiguity Diagnosis for Terms in Digital Humanities. In *Language Resources and Evaluation Conference*, Portorož, Slovenia.

Hervé Déjean. 2007. *pdf2xml open source software*. Last access on July 31, 2019.

Hervé Déjean and Jean-Luc Meunier. 2005. Structuring documents according to their table of contents. In *Proceedings of the 2005 ACM symposium on Document engineering*, pages 2–9.

Hervé Déjean and Jean-Luc Meunier. 2006. A system for converting pdf documents into structured xml format. In *Document Analysis Systems VII*, pages 129–140, Berlin, Heidelberg. Springer Berlin Heidelberg.

Hervé Déjean and Jean-Luc Meunier. 2010. Reflections on the inex structure extraction competition. In *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*, DAS '10, page 301–308, New York, NY, USA. Association for Computing Machinery.

Antoine Doucet and Miro Lehtonen. 2007. Unsupervised classification of text-centric xml document collections. In *Comparative Evaluation of XML Information Retrieval Systems, Fifth International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2006*, volume 4518 of *Lecture Notes in Computer Science*, pages 497–509. Springer.

Ismail El Maarouf, Juyeon Kang, Abderrahim Aitazzi, Sandra Bellato, Mei Gan, and Mahmoud El-Haj. 2021. The Financial Document Structure Extraction Shared Task (FinToc 2021). In *The Third Financial Narrative Processing Workshop (FNP 2021)*, Lancaster, UK.

Corina Florescu and Cornelia Caragea. 2017. PositionRank: An unsupervised approach to keyphrase extraction from scholarly documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1115, Vancouver, Canada. Association for Computational Linguistics.

Emmanuel Giguet and Gaël Lejeune. 2021. Daniel at the FinSBD-2 task : Extracting Lists and Sentences from PDF Documents: a model-driven end-to-end approach to PDF document analysis. In *Second Workshop on Financial Technology and Natural Language Processing in conjunction with IJCAI-PRICAI 2020*, Proceedings of the Second Workshop on Financial Technology and Natural Language Processing, pages 67–74, Kyoto, Japan.

Charlotte Lecluze and Gaël Lejeune. 2014. Deft2014, automatic analysis of literary and scientific texts in french (deft 2014, analyse automatique de textes littéraires et scientifiques en langue française)[in french]. In *TALN-RECITAL 2014 Workshop DEFT 2014: DÉfi Fouille de Textes (DEFT 2014 Workshop: Text Mining Challenge)*, pages 11–19.

Gaël Lejeune, Romain Brixtel, Charlotte Lecluze, Antoine Doucet, and Nadine Lucas. 2013. Added-value of automatic multilingual text analysis for epidemic surveillance. In *Artificial Intelligence in Medicine (AIME)*, pages 284–294.

Simone Marinai, Emanuele Marino, and Giovanni Soda. 2010. Table of contents recognition for converting pdf documents in e-book formats. In *Proceedings of the 10th ACM symposium on Document engineering*, pages 73–76.