Daniel at the FinSBD-2 task : Extracting Lists and Sentences from PDF Documents: a model-driven end-to-end approach to PDF document analysis

Emmanuel Giguet[1*] and **Gaël Lejeune**[2]
[1]Normandie Univ, UNICAEN, ENSICAEN, CNRS, GREYC
14000 Caen, France
[2]STIH, Sorbonne University
75005 Paris, France
emmanuel.giguet@unicaen.fr, gael.lejeune@sorbonne-universite.fr

Abstract

In this paper, we present the method we have designed and implemented for identifying lists and sentences in PDF documents while participating to FinSBD-2 Financial Document Analysis Shared Task. We propose a model-driven approach for the French and English datasets. It relies on a top-down process from the PDF itself in order to keep control of the workflow. Our objective is to use PDF structure extraction to improve text segment boundaries detection in an end-to-end fashion.

1 Introduction

Our team participated to the FinSBD-2 Shared Task dedicated to Sentence Boundary Detection in Financial Prospectuses. The task aims at identifying sentences, ordered lists, unordered lists, and list items in PDF Documents. It also aims at recovering the hierarchical structure of embedded lists. It was our first participation to this shared task. Our motivation is to improve our model-driven approach to multilingual document analysis.

Our approach was illustrated in FinTOC'2019 Shared Task [Rémi Juge, 2019]. It was dedicated to Table Of Content structure extraction from PDF Financial Documents. The task aimed at identifying and organizing the headers of the document according to its hierarchical structure. Since our approach gave good results on this task [Giguet and Lejeune, 2019], we took advantage of the second edition of FinTOC, called FinTOC-2020, to improve the implementation of our model driven approach. We seized the opportunity of the second edition of FinSBD [Ait Azzi *et al.*, 2019], to promote our overall model-driven approach and to enrich our document model with smaller units, i.e paragraphs, sentences and lists.

The paper is organized as follows. Section 2 discusses the rationale behind the interest for sentences in NLP. Section 3 provides the state of the art approaches to tackle the problem. Section 4 presents the preprocessing applied to the PDF documents. The next sections are dedicated to our document model-based approach: section 5 presents how to handle page layout at document scope; section 6 describes the detection and Structure Induction of various Document Objects including headers and footers, tables, lists, paragraphs. Section 7 presents the results we obtained on the task. Section 8 concludes and gives some perspectives about this work.

2 The importance of sentences in NLP Architectures

One way to present NLP tasks is to describe them as a series of transformation from an input format (e.g., a PDF file, a HTML file, a set of character strings) to an output format suitable for downstream components (e.g., an XML or JSON enriched file) or for end-users (e.g., a HTML file). NLP Components piped together form an NLP pipeline.

The rationale behind the NLP pipeline is to favor factorization and reuse of existing NLP components. In that way, tackling a new task may consist in preprocessing a new input format in order to feed an existing NLP pipeline (e.g., converting PDF binary file to machine-readable text) or to compose a pipeline by choosing the appropriate components for a given task. In NLP, the two main input and output formats are inherited from traditional grammar: words and sentences. The text representation is usually reconstructed from these units (see for instance DOC2VEC [Le and Mikolov, 2014]).

For many NLP components, the word (approximated by the concept of "token") is the core analysis grain and the sentence is the parsing frame. In that perspective, solving the tokenization task is considered to be a prerequisite to the proper application of NLP techniques. In some languages the task will be considered to be solved, in particular for English [Smith, 2020], while in under resourced languages like dialects [Bernhard *et al.*, 2017] or ancient language variants [Gabay *et al.*, 2019] the task requires intensive care from the research community. Tokens and sentences can be produced from any character string but it is common to tokenize from small text blocks (e.g., paragraphs, headers) rather than complete documents. Text blocks are search spaces in which token boundaries and sentence boundaries are computed. To this end, text blocks are expected to be large enough to contain at least one sentence. It should not be too large to limit memory consumption and computational costs.

*Contact Author

While many algorithms can handle long text blocks the same way they process short blocks, some NLP approaches are very sensitive to the input length. Some tasks can be performed by linear time algorithms, other ones involve a higher computational cost, with a time complexity sometimes higher than quadratic in some cases [Corro, 2020]. Another example is word embeddings models where the input length is a key feature. For instance, the CAMEMBERT model (BERT for French) can not process inputs longer than five hundred twelve tokens [Martin *et al.*, 2020]. Obviously, whatever the time complexity, the space complexity also has to be controlled in order to limit the amount of space and memory taken by the underlying algorithms.

Processing long documents such as financial prospectuses with an NLP pipeline requires the proper definition of text blocks and sentences. While small text blocks such as paragraphs or headers are expected to be easily computed, the NLP architect has to guarantee that the computed blocks and sentences always fit the requirement of the different NLP components, in terms of time and space complexity. The task is not trivial when the pipeline is made of NLP components designed by various contributors.

Assuming that paragraphs and headers are always relevant text blocks leads to failure. It does not guarantee that paragraphs will never exceed the expected maximum length. Similarly, assuming that sentence tokenization guarantees a certain maximum length fails when confronted to text without any punctuation.

3 Sentence Segmentation in Practice

3.1 State of the art on Sentence Boundary Detection

Sentence boundary detection plays a crucial role in Natural Language Processing. For a long time, sentences have been considered as given input. Things changed with the rise of robust parsing in the 90's. What is a sentence and how to detect them automatically in raw texts becomes crucial. The concept of sentence is questioned by [Grefenstette and Tapanainen, 1994]. [Giguet, 1995] adresses the problem in the context of multilingual sentence tokenization in raw texts.

[Ait Azzi *et al.*, 2019] retraces the milestones and achievements in Sentence Boundary Detection, from the first rule-based approaches to the recent deep-learning approaches. They open a new challenge related to the identification of sentences in noisy unstructured PDF documents from the finance sector. Obviously the issue is much broader and impacts all the works concerning PDF document analysis.

[Dale *et al.*, 2000] highlights the fact that there are four challenges for sentence segmentation: (I) Language Dependence, (II) Character-Set Dependence, (III) Application Dependence and (IV) Corpus Dependence. Except for "Application Dependence", all of these challenges are linked to variations in the input data. Therefore, the techniques that have been developed by the community tend to focus on handling the variability of the data to be processed. One of the main questions regarding SBD is whether the sentence boundaries are explicitly marked or not. It relies to both language dependence and corpus dependence since all languages will not mark explicitly depending on the text genre for instance. In an experiment on multilingual SBD [Kiss and Strunk, 2006] it has been shown that the main issue is that the period can serve multiple purposes and that handling this "polysemy" allows to get rid of most of boundary errors. In other application domains there has been more focus on the detection of sentence starters, obviously the best example would be speech processing [Bachenko *et al.*, 1995].

3.2 State of the art on Lists and Enumerations

Research on lists and enumerations as text objects playing an important role in the text architecture has been conducted by [Virbel, 1999; Pascual and Virbel, 1996] in Toulouse, France. [Luc, 2001] studied the representation of the internal structure of enumerations with two text structure models: the Rhetorical Structure Theory (RST) and the model of text architecture, dedicated to the study and representation of visuo-spatial structures of texts. [Maurel, 2004; Maurel *et al.*, 2006] studied visuo-spatial structures of texts, in particular enumerations. This work, related to Natural Language Processing, concerns the oral transposition of these structures by Text-To-Speech systems. Regarding list detection, [Déjean, 2010] introduced a method for detecting numbered sequence in documents.

4 Preprocessing PDF Documents

In previous INEX Book Structure Extraction Competitions, we used to consider the whole document to extract the structure [Giguet and Lucas, 2010a; Giguet and Lucas, 2010b; Giguet *et al.*, 2009]. In our participation to this FinSBD shared task; we wanted to start over from this approach in order to get an end-to-end pipeline from the PDF file itself to sentence segmentation[1]. The experiment is conducted from PDF documents to ensure the control of the entire process. The document content is extracted using the `pdf2xml` command [Déjean, 2007]. It allow us to extract text content with its structural information via vectorial shapes.

4.1 Dealing with Text Content : tokens, lines and blocks

There is no concept of "word" or "number" or "token" in a PDF file. Therefore, these units have to be inferred. In order to ease the processing, `pdf2xml` defines a "token" as a computational unit based on character spacing. In practice, most output tokens correspond to words or numbers but they can also correspond to a composition of several interpretable unit (e.g., "Introduction . . . . . 5" or a breakdown of an interpretable unit (e.g., "C" "O" "N" "T" "E" "N" "T").

We assume that the PDF financial prospectuses are automatically generated by the PDF converter of a word processor. Thus, we do not check if the document is a scanned document or if it is the output of an OCR application.

Consequently, we do not consider possible trapezoid or parallelogram distortion, page rotation or curved lines. This assumption simplifies the initial stages: baselines and linespacing are inferred from the coordinates on the y-axis; left,

[1]Code : https://github.com/rundimeco/daniel_fintoc2019/FINSBD2/

right and centered alignments are inferred from the coordinates on the x-axis.

For `pdf2xml`, tokens are linked to two other units that we do not use in our experimentation:

line : a sequence of tokens which may correspond to a coherent visual text line (relatively to token-spacing)

block : a sequence of lines which may correspond to a coherent visual text block (relatively to line-spacing).

We only rely on the `pdf2xml` "token" unit. We redefine our own "line" unit in order to better control the coherence of our hierarchy of graphical units.

4.2 Dealing with Vectorial Shapes

One of the main advantages of using `pdf2xml` is to enable our approach to rely on vectorial information during document analysis. Text background, framed content, underline text, table grid are crucial information that contributes to sense making: we have no reason to ignore them. They simplify the reader's task of sense-making [Sorin, 2015], so that they may 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 enables to build composite vectorial shapes and to interpret them as text backgrounds, cell borders, underlines. This preprocessing component returns results that are used to detect framed content and table grids.

As an example of vectorial preprocessing, more than thirteen rectangles may have to be processed to identify a single table cell with background and borders:

- eight adjacent rectangles for the borders : two horizontal borders, two vertical borders and four square corners;
- at least five rectangles for paving the cell background: four rectangles for cell paddings, and at least one rectangle for text background.

In the context of FinSBD, dealing with vectorial information allows to detect tables and ignore them. Since no sentence boundaries has to be searched in tables, table exclusion contributes to avoid the generation of potentially numerous false positive since tables are long and frequent in financial prospectuses.

5 Handling Page Layout at Document Scope

5.1 Dealing with Content Areas

We assume that PDF converters serialize the content of a page area by area, depending on the page layout. A *content area* corresponds to a page subdivision such as a column, a header, a footer, or a floating table or figure. However, content areas are represented neither in the PDF structure nor in the `pdf2xml` output. Content areas are implicitly inferred and interpreted by the cognitive skills of the reader.

As an example, the repetition of a content located at the bottom of contiguous pages (i.e., positional information), with identical style properties (i.e., morphological information), visually detached from the above content and smaller than the above content (i.e., contrastive information), leads the reader to perceive a content area and to interpret it as a footer. In the PDF document, it is only a series of characters with style properties in the 2D coordinate system of sequential pages.

When a content area is processed by the PDF converter, we assume that characters and lines are serialized in reading order, so that there is no ordering problem to consider inside a content area. However, when parsing a page, we cannot always expect to find the content serialized in reading order: PDF converters can serialize content areas in several ways. For instance, the header and footer areas can be serialized before the page's main content. Indeed, the boundary delimitation of content areas inside a page is one of the main challenges.

Bounding the content areas over pages is not immediate due to the absence of marks that separate them from other adjacent areas. In our process, positional information, morphological information and contrastive information are inferred from the document structure in order to help the boundary delimitation of content areas.

5.2 From Page Layout to Page Layout Models

Page Layout Analysis (PLA) aims at recognizing and labeling content areas in a page (e.g., text regions, tables, figures, lists). It is the subject of abundant research and articles. ICDAR challenges show the efforts of a large international community interested in Document Analysis and Recognition [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. This strategy has been used earlier, in the Resurgence Project [Giguet, 2008; Giguet, 2011].

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 receive particular content 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.

5.3 Detecting Header and Footer Areas

Header area boundaries 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.

Considering FinSBD Task, identifying header and footer allows to build the main content flow over pages. Hence, it avoids to get paragraphs, sentences or list items merged with header and footer content when they overlap two pages.

6 Parsing with a Document Model

In INEX Book Structure Extraction Competition, we introduced a relevant strategy to divide a document in main parts and chapters [Giguet *et al.*, 2009].

As we participated at FinTOC Shared Task [Giguet and Lejeune, 2019], we used a fallback strategy to divide the document in parts: the Table of Content (TOC) if detected is used to separate preliminaries from the main content of the document. The underlying idea is to rely on the main part's internal regularities when making decisions. This is useful for inferring paragraph models or list item models.

Contrary to our model-based approach, this fallback does not allow to separate the document body from its appendices or annexes. That is unfortunate since appendices and annexes may have their own regularities that should not be mixed with the document body regularities in the inference engine.

6.1 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 5.3). Then header is searched in the related PDF page. When found, the corresponding line is categorized as header.

6.2 Detection and Structure Induction of Document Objects in PDF Documents

In the main content area of our model, column areas and floating areas are both planned to contain information. While column areas are planned to contain the main text stream, floating areas are planned to contain spotlight contents that are relatively independent from the main content. Floating areas contains information that are not part of the main stream of text. Figures, tables, framed text may be such autonomous document components.

6.3 Table Detection and Table Structure Induction in PDF Documents

Table detection and table structure induction are beyond the scope of this article. However table detection is important for FinSBD in order to exclude table content from the main text stream. This way, we are able to exclude table rows when searching for list items or sentences. Table structure induction does not affect list or sentence boundary detection modules.

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.

The table structure is inferred from the vectorial grid, the vectorial cell backgrounds, and the inner text spacing. This way we handle table cells that span over multiple columns. Due to lack of time, table cells that span over multiple rows is not implemented yet.

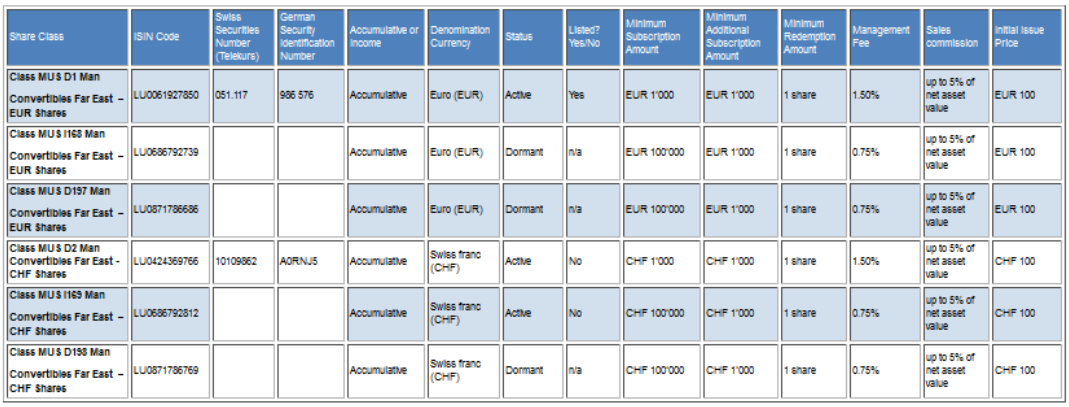
Table detection and table structure induction have been designed and implemented outside FinSBD and reused as is for convenience.

6.4 Unordered List Structure Induction in PDF Documents

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

Unordered list items are searched at page scope. The typographical symbols (i.e., glyph) used to introduce items are not predefined. We infer the symbol by identifying multiple left-aligned lines introduced by the same single-character token. This strategy allows the algorithm to capture various bullet symbols such as squares, white bullets... Alphabetical or decimal characters are rejected as possible bullet style type.

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.



Share Class	ISIN Code	Swiss Securities Number (Telekurs)	German Security Identification Number	Accumulative or Income	Denomination Currency	Status	Listed? Yes/No	Minimum Subscription Amount	Minimum Additional Subscription Amount	Minimum Redemption Amount	Management Fee	Sales commission	Initial Issue Price
Class MU$ D1 Man Convertibles Far East – EUR Shares	LU0061927850	051.117	986 576	Accumulative	Euro (EUR)	Active	Yes	EUR 1'000	EUR 1'000	1 share	1.50%	up to 5% of net asset value	EUR 100
Class MU$ I168 Man Convertibles Far East – EUR Shares	LU0686792739			Accumulative	Euro (EUR)	Dormant	n/a	EUR 100'000	EUR 1'000	1 share	0.75%	up to 5% of net asset value	EUR 100
Class MU$ D197 Man Convertibles Far East – EUR Shares	LU0871786686			Accumulative	Euro (EUR)	Dormant	n/a	EUR 100'000	EUR 1'000	1 share	0.75%	up to 5% of net asset value	EUR 100
Class MU$ D2 Man Convertibles Far East - CHF Shares	LU0424369766	10109862	A0RNJ5	Accumulative	Swiss franc (CHF)	Active	No	CHF 1'000	CHF 1'000	1 share	1.50%	up to 5% of net asset value	CHF 100
Class MU$ I169 Man Convertibles Far East – CHF Shares	LU0686792812			Accumulative	Swiss franc (CHF)	Active	No	CHF 100'000	CHF 1'000	1 share	0.75%	up to 5% of net asset value	CHF 100
Class MU$ D198 Man Convertibles Far East – CHF Shares	LU0871786769			Accumulative	Swiss franc (CHF)	Dormant	n/a	CHF 100'000	CHF 1'000	1 share	0.75%	up to 5% of net asset value	CHF 100

Figure 1: Illustration of a PDF table rendered as a HTML/CSS table thanks to vectorial shape analysis

6.5 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 analyze each numbering prefix as a tuple $\langle P, S, I, C \rangle$ where :

P refers to the numbering pattern (string);

S : numbering style type (single character, see below);

I : numbering count written in numbering style type (single character);

C : decimal value of the numbering count (integer).

The numbering style types are defined as follows :

- Unambiguous style types:
  - D: Decimal
  - L: Lower-Latin
  - M: Upper-Latin
  - G: Lower-Greek
  - H: Upper-Greek
  - R: Lower-Roman
  - S: Upper-Roman
- Ambiguous style types:
  - ?: Lower-Latin OR Lower-Roman
  - !: Upper-Latin OR Upper-Roman

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

Then, lines are grouped in clusters sharing the same numbering pattern, for instance:

- 2.a and 2.b → cluster 2.L (Lower-Latin)
- A.2.c) and A.2.f) → cluster A.2.L (Lower-latin)
- A.2.i) and A.2.v) → cluster A.2.? (ambiguous, Lower-Latin or Lower-Roman)

The disambiguation process separates ambiguous line clusters from unambiguous line clusters. Ambiguous patterns are mapped to their corresponding unambiguous patterns. For instance, A.2.?) is mapped to A.2.L) and A.2.R) if patterns exist.

The disambiguation process assigns an unambiguous style type to ambiguous lines. The process relies on compatible unambiguous clusters as disambiguation contexts.

Two cases are considered:

1. Ambiguous lines that are mapped to a single unambiguous patterns are directly disambiguated. For instance, A.2.?) is directly mapped to A.2.L) if no cluster A.2.R) exists.
2. Ambiguous lines that can be mapped to multiple unambiguous patterns are analyzed to identify a compatible unambiguous cluster. For instance, line A.2.v) is compatible with the cluster A.2.R) if A.2.v) is missing in the cluster and if line numbering A.2.iv) exists and both line numbers and/or left-alignments are compatibles.

Once the disambiguation stage is achieved, we split every cluster in order to build ordered series. For instance, the cluster containing lines 2.a, 2.b, 2.c, 2.a, 2.b is split in two ordered series 2.a, 2.b, 2.c and 2.a, 2.b.

Finally, we detect and resolve missing or unexpected items. For instance, first item of a numbered list may be missing when the numbering item is located on the same line of its parent item (the missing item is the second token): list item (a) is not detected when line starts with (2) (a). For instance, (X) is an unexpected item which must be removed from the cluster: (A), (B), (C), (X), (D), (E).

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

6.6 Paragraph Structure Induction in PDF Documents

The aim of paragraph structure induction is to infer paragraph models that are later used to detect paragraph instances. A

paragraph model can be seen as a paragraph style defined in any word processor (see figure 2).

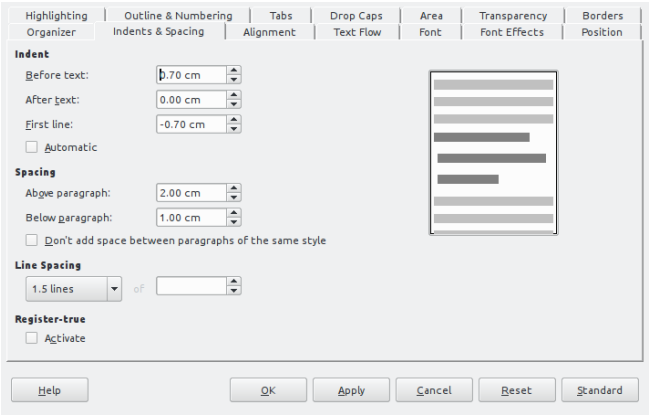


Figure 2: Settings in paragraph style of LibreOffice word processor

In other words, the aim of the process is 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 no 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. Paragraph bodies are 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.

Reliable paragraph lines are categorized as follows: L for leading line, B for body lines, T for trailing line. Header lines are categorized H (see section 6.1). 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 (see sections 6.4 and 6.5 above).

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.

7 Evaluation of the Sentence Boundary Detection

While the shared task is dedicated to Sentence Boundary Detection, we focused on designing our top-down pipeline for the PDF itself. Therefore, we did not have enough time to fine tune the output. The main difficulty we encountered was to align our internal representation to the expected FinSBD representation since both representations are very different. A complex ad-hoc module had to be implemented to try to map our structure to the expected character-based structure.

Our algorithm consists in splitting the text blocks extracted by our top-down pipeline using a single regular expression based on the presence of an end of sentence punctuation mark followed by a space separator or a line separator. In the following tables we show the detailed results of an improved version of our system in which the beginning and end of paragraphs are correctly detected. We only kept the subtask1 result of our original submission to ease comparisons. We removed the results on lists and numbered items since our system does not give these units yet.

In Table 1 and table 2 are shown the results obtained on the train set, respectively in English and in French. We focused on the sentence and the items for the system we submitted. Our system has much better results in terms of Precision but seems to miss many sentences.

Document	sent			item		
	f1	prec.	recall	f1	prec.	recall
Invesco-Fu	37.8	44.2	33.0	0	0.0	0
EdR-Privat	43.7	34.8	58.9	11.1	78.6	6.0
CANDRIAM-G	63.8	83.7	51.5	78.5	73.9	83.6
Dexia-Equi	65.9	80.5	55.8	46.1	67.3	35.0
Credit-Sui	78.5	89.9	69.8	48.0	69.1	36.7
Macro	57.9	66.6	53.8	36.7	57.8	32.3

Table 1: Results on the English train set, 32.6 F-measure on subtask1 (VS 23.6 for our official submission) sorted by F-measure on sentences

Our results on the test set are shown in Table 3 and table 4. One can see that the results are high in English as compared to the train set but the dataset is too small to draw any conclusion from that. The fact that the same pattern in French maybe show that our rule based system does not suffer too much from over-fitting.

8 Conclusion

Our team participated to the FinSBD-2 Shared Task dedicated to Sentence Boundary Detection in Financial Prospectuses. It was our first participation to this shared task. Our motivation was to improve our model driven approach to multilingual document analysis.

The work we have achieved is very promising. We had the opportunity to handle the full workflow and to define, control and implement each NLP component.

Concerning FinSBD shared task, we lack time to finalize the creation of list objects, unordered list objects and sentences. We chose to control the whole workflow and it was a bit too ambitious regarding time constraints since aligning our internal representations to the offsets of the groundtruth.

In a near future, we intend to enhance the implementation of our page layout model in order to be compliant with the page layout model described in [Giguet, 2008]. We would also like to implement the document model we introduced in INEX Book Structure Extraction Competition in order to divide a document in main parts and chapters [Giguet *et al.*, 2009]. This strategy applied at document scope could have

Document		sent			item	
	f1	prec.	recall	f1	prec.	recall
LCL-OBLIGA	28.8	36.3	23.8	2.9	3.4	2.6
LCL-DOUBLE	33.4	36.8	30.7	3.7	5.3	2.9
LCL-INVEST	34.6	43.6	28.7	1.1	2.6	0.7
AMUNDI-VIE	34.9	44.3	28.8	2.5	6.5	1.6
FUNDQUEST-	38.1	51.9	30.1	43.0	44.4	41.7
BNP-PARIBA	44.8	70.8	32.8	45.0	39.1	52.9
QUILVEST-C	51.0	62.1	43.3	34.6	51.9	25.9
GROUPAMA-O	53.1	60.5	47.3	39.7	40.0	39.5
AVIVA-INTE	53.3	66.1	44.6	32.0	29.1	35.6
CREDIT-MUT	53.7	83.5	39.6	33.8	26.1	47.9
GUTENBERG-	54.2	58.4	50.5	34.5	37.0	32.3
Fondo-BNP-	57.2	59.2	55.3	66.7	73.7	60.9
CM-CIC-EUR	57.3	56.8	57.8	44.8	41.4	48.8
FCPI-IDINV	59.8	78.4	48.3	88.9	88.9	88.9
GASPAL-CON	61.5	73.4	52.9	35.8	70.7	24.0
Le-PALÉ-FR	62.0	76.8	51.9	60.1	48.8	78.2
NORDEN-SMA	62.1	74.0	53.4	49.7	40.4	64.7
ORCHIDEE-I	62.3	64.8	60.1	54.5	70.6	44.4
SÉLECT-OBL	65.6	90.9	51.3	32.9	28.6	38.7
Sécuri-Tau	68.1	84.3	57.1	28.6	19.0	57.1
QUADRIGE-M	69.2	85.6	58.1	82.8	89.1	77.4
FCPI-Innov	72.8	84.6	63.9	50.2	52.6	48.1
INNOVEN-EU	77.7	89.7	68.5	8.5	11.8	6.7
Macro	54.6	66.6	46.9	38.1	40.0	40.1

Table 2: Results on the French train set 31.9 F-measure on subtask1 (VS 33.5% for our original submission) sorted by F-measure on sentences

Document		sent			item	
	f1	prec.	recall	f1	prec.	recall
Arabesque-	71.8	88.4	60.5	55.3	88.7	40.1
MAGALLANES	76.9	92.1	66.0	23.8	88.9	13.7
Macro	74.3	90.2	63.2	39.5	88.8	26.9

Table 3: Results on the English test set : 37.9 F-measure on subtask1 (VS 31.7 for our original submission) sorted by F-measure on sentences

made more accurate decisions at lower level of the hierarchy (i.e., divide-and-conquer strategy).

References

[Ait Azzi *et al.*, 2019] Abderrahim Ait Azzi, Houda Bouamor, and Sira Ferra. The finsbd-2019 shared task:sentence boundary detection in pdf noisy text in the financial domain. In Chung-Chi Chen, Hen-Hsen Huang, Hiroya Takamura, and Hsin-Hsi Chen, editors, *Proceedings of the First Workshop on Financial Technology and Natural Language Processing*, pages 74–80, Macao, China, August 2019.

[Antonacopoulos *et al.*, 2009] Apostolos Antonacopoulos, David Bridson, Christos Papadopoulos, and Stefan Pletschacher. 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, 01 2009.

Document		sent			item	
	f1	prec.	recall	f1	prec.	recall
CM-CIC-OBL	32.3	27.7	38.8	40.0	36.2	44.7
LCL-MULTI-	35.3	34.5	36.2	0	0.0	0.0
HEXASTEP-H	40.6	71.5	28.3	11.1	33.3	6.7
AMUNDI-IND	50.0	51.5	48.6	0	0	0.0
LAZARD-ACT	57.1	69.8	48.2	39.1	29.3	58.6
FIP-IXO-DE	58.7	71.1	50.0	50.0	100.0	33.3
BNP-Pariba	59.0	54.1	64.9	31.6	24.5	44.4
GREEN-BOND	59.5	67.1	53.6	27.6	52.2	18.8
KLE-EONIA-	60.4	70.3	53.0	63.5	61.4	65.9
ECUREUIL-P	67.8	73.0	63.4	55.7	53.1	58.6
Macro	52.1	59.1	48.5	31.9	39.0	33.1

Table 4: Results on the French test set, 27.98 F-measure on subtask1 (VS 26.2 for our original submission) sorted by F-measure on sentences

[Bachenko *et al.*, 1995] Joan Bachenko, Eileen Fitzpatrick, and Jeffrey Daugherty. A rule-based phrase parser for real-time text-to-speech synthesis. *Natural Language Engineering*, 1(2):191–212, 1995.

[Bernhard *et al.*, 2017] Delphine Bernhard, Amalia Todirascu, Fanny MARTIN, Pascale Erhart, Lucie Steible, Dominique Huck, and Christophe Rey. Problèmes de tokénisation pour deux langues régionales de France, l'alsacien et le picard. In *DiLiTAL 2017*, Actes de l'atelier " Diversité Linguistique et TAL ", pages 14–23, Orléans, France, June 2017.

[Corro, 2020] Caio Corro. Span-based discontinuous constituency parsing: a family of exact chart-based algorithms with time complexities from o(n^6) down to o(n^3), 2020.

[Dale *et al.*, 2000] Robert Dale, H. L. Somers, and Hermann Moisl. *Handbook of Natural Language Processing*. Marcel Dekker, Inc., USA, 2000.

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

[Déjean and Meunier, 2010] Hervé Déjean and Jean-Luc Meunier. 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, 2010. Association for Computing Machinery.

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

[Déjean, 2010] Hervé Déjean. Numbered sequence detection in documents. In Laurence Likforman-Sulem and Gady Agam, editors, *Document Recognition and Retrieval XVII*, volume 7534, pages 41 – 52. International Society for Optics and Photonics, SPIE, 2010.

[Gabay *et al.*, 2019] Simon Gabay, Marine Riguet, and Loïc Barrault. A Workflow For On The Fly Normalisation Of 17th c. French. In *DH2019*, Utrecht, Netherlands, July 2019. ADHO.

[Giguet and Lejeune, 2019] Emmanuel Giguet and Gaël Lejeune. Daniel@FinTOC-2019 shared task : TOC extraction and title detection. In *Proceedings of the Second Financial Narrative Processing Workshop (FNP 2019)*, pages 63–68, Turku, Finland, September 2019. Linköping University Electronic Press.

[Giguet and Lucas, 2010a] Emmanuel Giguet and Nadine Lucas. The book structure extraction competition with the resurgence software at caen university. In Shlomo Geva, Jaap Kamps, and Andrew Trotman, editors, *Focused Retrieval and Evaluation*, pages 170–178, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[Giguet and Lucas, 2010b] Emmanuel Giguet and Nadine Lucas. The book structure extraction competition with the resurgence software for part and chapter detection at caen university. In Shlomo Geva, Jaap Kamps, Ralf Schenkel, and Andrew Trotman, editors, *Comparative Evaluation of Focused Retrieval - 9th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2010, Vugh, The Netherlands, December 13-15, 2010, Revised Selected Papers*, volume 6932 of *Lecture Notes in Computer Science*, pages 128–139. Springer, 2010.

[Giguet *et al.*, 2009] Emmanuel Giguet, Alexandre Baudrillart, and Nadine Lucas. Resurgence for the book structure extraction competition. In Shlomo Geva, Jaap Kamps, and Andrew Trotman, editors, *INEX 2009 Workshop Pre-Proceedings*, pages 136–142, 2009.

[Giguet, 1995] Emmanuel Giguet. Multilingual sentence categorization according to language. In *Proceedings of the European Chapter of the Association for Computational Linguistics (EACL) SIGDAT Workshop "From text to tags : Issues in Multilingual Language Analysis*, pages 73–76, March 1995.

[Giguet, 2008] Emmanuel Giguet. Rapport scientifique du projet résurgence. Technical report, Université de Caen Basse-Normandie, November 2008.

[Giguet, 2011] Emmanuel Giguet. *De l'analyse syntaxique automatique à l'analyse automatique du discours dans les collections multilingues de documents numériques composites*. Mémoire d'habilitation à diriger des recherches, Université de Caen Basse-Normandie, September 2011.

[Grefenstette and Tapanainen, 1994] Gregory Grefenstette and Pasi Tapanainen. What is a word, what is a sentence? problems of tokenization. In *Proceedings of the 3rd Conference on Computational Lexicography and TextResearch*, pages 79–87, 1994.

[Kiss and Strunk, 2006] Tibor Kiss and Jan Strunk. Unsupervised multilingual sentence boundary detection. *Computational Linguistics*, 32(4):485–525, 2006.

[Le and Mikolov, 2014] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. *CoRR*, abs/1405.4053, 2014.

[Luc, 2001] Christophe Luc. Une typologie des énumérations basée sur les structures rhétoriques et architecturales du texte. In *TALN2001, Université de Tours, 05/07/2001-07/07/2001*, pages 263–272. ., juillet 2001.

[Martin *et al.*, 2020] Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric Villemonte de la Clergerie, Djamé Seddah, and Benoît Sagot. Camembert: a tasty french language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

[Maurel *et al.*, 2006] Fabrice Maurel, Mustapha Mojahid, Nadine Vigouroux, and Jacques Virbel. Documents numériques et transmodalité. transposition automatique à l'oral des structures visuelles de texte. *Document numérique*, 9, 09 2006.

[Maurel, 2004] Fabrice Maurel. *Transmodalité et multimodalité écrit/oral : modélisation, traitement automatique et évaluation de stratégies de présentation des structures "visuo-architecturale" des textes*. PhD thesis, Université de Toulouse 3, 2004.

[Pascual and Virbel, 1996] Elsa Pascual and Jacques Virbel. Semantic and Layout Properties of Text punctuation. 34th Annual meeting of the Association for Computational Linguistics. In *International Workshop on Punctuation in Computational Linguistics, Santa Cruz, USA,* , Santa Cruz, USA, juin 1996. Univ. of California. Dates de conférence : juin 1996 1996. Pages de la publication : ?.

[Rémi Juge, 2019] Sira Ferradans Rémi Juge, Najah-Imane Bentabet. The fintoc-2019 shared task: Financial document structure extraction. In *The Second Workshop on Financial Narrative Processing of NoDalida 2019*, 2019.

[Smith, 2020] Noah A. Smith. Contextual word representations: Putting words into computers. *Commun. ACM*, 63(6):66–74, May 2020.

[Sorin, 2015] Laurent Sorin. *Contributions of textual architectures to the non-visual accessibility of digital documents*. Theses, Université Toulouse le Mirail - Toulouse II, December 2015.

[Virbel, 1999] Jacques Virbel. Structures textuelles. Planches. Fasciccule I : Enumérations. Rapport de recherche -, IRIT, Université Paul Sabatier, Toulouse, février 1999.