Proceedings of the Seventh International Conference on Research in Intelligent and Computing in Engineering pp. 151–156 DOI: 10.15439/2022R26 ISSN 2300-5963 ACSIS, Vol. 33

Improving Logical Structure Analysis of Visually Structured Documents with Textual Features

Huu-Loi Le*, Nghia Luu Trong†, Huyen Ngo Thanh‡
*Hung Yen University of Technology and Education, Hung Yen, Vietnam
Email: lehuuloi.cs@gmail.com
†Hanoi University of Science and Technology, Hanoi, Vietnam
Email: nghia.lt204888@sis.hust.edu.vn
‡Hung Yen University of Technology and Education, Hung Yen, Vietnam
Email: nthuyen@utehy.edu.vn
‡Corresponding author

***Abstract*—This paper introduces a new model to improve the quality of logical structure analysis of visually structured documents. To do that, we extend the model of Koreeda and Manning [1]. In order to enhance textual features, we define a new feature that uses the font size of texts as an indicator. As our observation, the font size is an important indicator that can be used to represent the structure of a document. The new font size feature is combined with visual, textual, and semantic features for training an analyzer. Experimental results on four legal datasets show that the new font size feature contributes to the model and helps to improve the F-scores. The ablation study also shows the contribution of each feature in our model.**

***Index Terms*—Logical structure analysis, VSDs, feature engineering, information extraction.**

I. Introduction

A lot of natural language processing (NLP) models, tasks, and pipelines usually require clean texts for training and processing. However, in real applications, raw data may be not clean and well organized. For example, legal documents (e.g., contracts and legal codes) are not so clean and many documents use visually structured documents (VSDs) such as PDFs [1]. As pointed out by Obermaier et al. [2], among 7.3 million documents found in Panama papers, approximately 30% were PDFs. Therefore, a good VSDs reader is required to facilitate NLP tasks in actual applications.

Compared to text data (e.g., news), VSDs contain richer information. For instance, Fig. 1 shows an example of the difference between a VSD and the raw text. While the raw text without logical structure analysis contains a sequence of tokens, the VSDsincludes both text and its structure. By reading the VSDs, we know which paragraphs are parents and which paragraphs are children. In addition, characteristics of the VSDs (e.g., font size, bold text, the position of a paragraph) can be taken into account as good indicators for logical structure analysis. We argue that an information extraction (IE) should be aware of the structure of a document to output high-quality extracted information, especially in specific domains (e.g., legal or business documents) [1], [3].

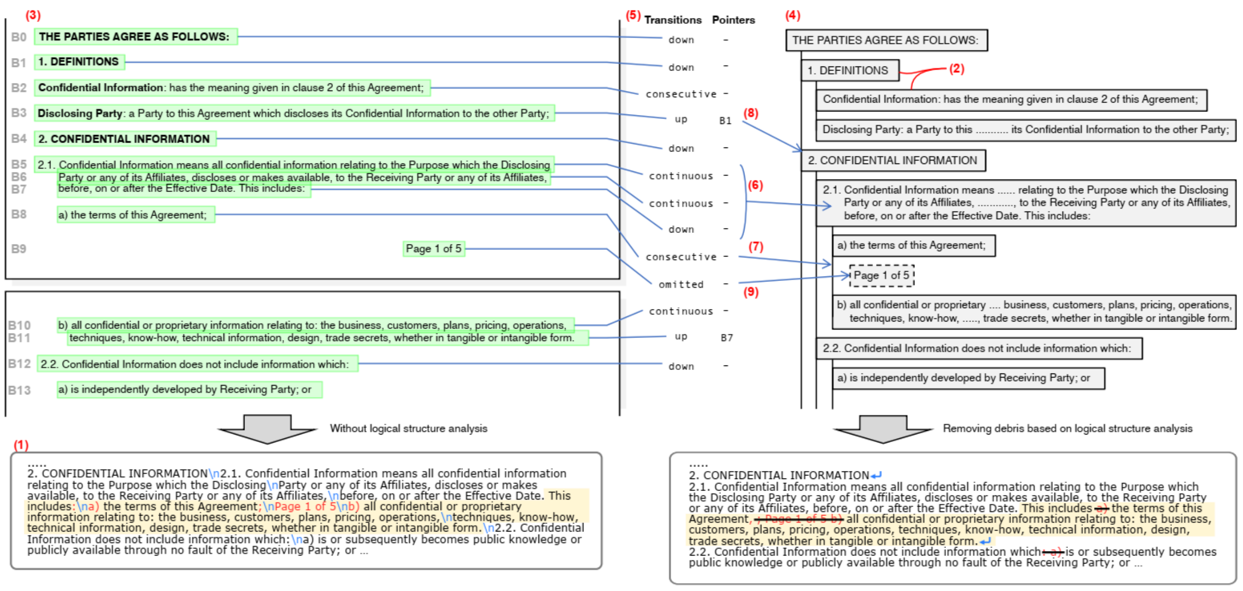
So far, there are two main directions for VSDs processing. The first direction uses rules for extracting the structure of a VSD [4]. The second direction is to analyze the structure of a VSD for logical structure parsing [5], [6] by using machine learning. While the rule-based approach can achieve high accuracy but it suffers from rule definition and management. On the other hand, the machine learning approach can generalize to new VSD types. Xu et al. introduced LayoutLM which can analyze the structure of a VSD [7], [8]. LayoutLM uses the Transformer architecture [9] to train a language model for VSDs. More recently, Koreeda and Manning introduce a model for capturing the logical structure of VSDs by using feature engineering [1]. The authors combined visual features, textual features, and semantic features for training a parser. The proposed model achieved good results on four legal datasets.

This paper improves the quality of logical structure analysis by using feature engineering. We extend the work of [1] to take into account visual, textual, and semantic features. For improvement, we also define a new feature that takes advantage of font size for training the model. Experimental on four types of datasets in the legal domain show that the proposed model obtains promising results compared to strong baselines. This paper makes two main contributions.

- It improves the quality of logical structure analysis by introducing a new feature that takes advantage of font size from texts. As our observation, the font size is an important indicator that can be used to represent the structure of a document. For example in Fig. 1, a title of a section has a larger font size than the title of its subsections. The new feature is combined with visual, textual, and semantic features to train the analyzer.
- It validates the contribution of the new feature and the proposed model on four datasets. Experimental results show that the model achieves promising results. The ablation study also shows the contribution of each feature that facilitates the next studies of logical structure analysis.

II. Related work

Hatsutori et al. [4] introduced a system that is based on the rule that fully depends on numberings. However, our idea and the result in Section IV can define that the system incorporating textual and semantic cues performs more effectively than their method. In contrast, *Sporleder and Lapata* [10]


(3)
B0 THE PARTIES AGREE AS FOLLOWS:
B1 1. DEFINITIONS
B2 Confidential Information: has the meaning given in clause 2 of this Agreement;
B3 Disclosing Party: a Party to this Agreement which discloses its Confidential Information to the other Party;
B4 2. CONFIDENTIAL INFORMATION
B5 2.1. Confidential Information means all confidential information relating to the Purpose which the Disclosing
B6 Party or any of its Affiliates, discloses or makes available, to the Receiving Party or any of its Affiliates,
B7 before, on or after the Effective Date. This includes:
B8 a) the terms of this Agreement;
B9 Page 1 of 5
B10 b) all confidential or proprietary information relating to: the business, customers, plans, pricing, operations,
B11 techniques, know-how, technical information, design, trade secrets, whether in tangible or intangible form.
B12 2.2. Confidential Information does not include information which:
B13 a) is independently developed by Receiving Party; or
(5) Transitions Pointers
down -
down -
consecutive -
up B1
down -
continuous -
continuous -
down -
consecutive -
omitted -
continuous -
up B7
down -
(4)
(2)
(6)
(7)
(8)
(9)
Without logical structure analysis
Removing debris based on logical structure analysis
(1)

.....
2. CONFIDENTIAL INFORMATION\n2.1. Confidential Information means all confidential information relating to the Purpose which the Disclosing\nParty or any of its Affiliates, discloses or makes available, to the Receiving Party or any of its Affiliates,\nbefore, on or after the Effective Date. This includes:\na) the terms of this Agreement;\nPage 1 of 5\nb) all confidential or proprietary information relating to: the business, customers, plans, pricing, operations,\ntechniques, know-how, technical information, design, trade secrets, whether in tangible or intangible form.\n2.2. Confidential Information does not include information which:\na) is or subsequently becomes public knowledge or publicly available through no fault of the Receiving Party; or ...

.....
2. CONFIDENTIAL INFORMATION↵
2.1. Confidential Information means all confidential information relating to the Purpose which the Disclosing Party or any of its Affiliates, discloses or makes available, to the Receiving Party or any of its Affiliates, before, on or after the Effective Date. This includes ~~a)~~ the terms of this Agreement, ~~; Page 1 of 5 b)~~ all confidential or proprietary information relating to: the business, customers, plans, pricing, operations, techniques, know-how, technical information, design, trade secrets, whether in tangible or intangible form.↵
2.2. Confidential Information does not include information which ~~: a)~~ is or subsequently becomes public knowledge or publicly available through no fault of the Receiving Party; or ...

Fig. 1: Sample of visualization of the logical structure analysis for VSDs [1].

proposed a system that fully depends on textual and semantic cues to detect a paragraph boundary for plain texts. Although their method is not focused on dealing with VSDs, we can incorporate their ideas as an additional feature of our system.

Abreu et al. [11] and *Ferrés et al.* [12] had their works in analysis of the logical structure by identifying some special structures in VSDs like subheading, ... These studies, however, are not handling in the paragraph-level logical structure and too coarse-grained. Therefore, those studies cannot satisfy our demands and fulfill our needs. *Xu et al.* [8] implemented more detailed and included extracted list items. Despite its improvement, it is still not suitable for our study because analysis of logical structures is not the center of this work.

Despite the difference in the goal, *Gillick* [13] proposed a sentence boundary detection system that has some similar textual features to our technique. But we apply richer textual and visual features that they do not employ to reach our goal which is to predict structures together with boundaries as precisely as possible.

A system proposed by *Koreeda et al.* [1] incorporates a combination of textual, visual, and semantic cues to analyze the logical structures in VSDs with a machine learning classifier. This work fully meets all our needs so we apply the same strategy for our research. In the proposed work, we improve classification performance by extracting and adding the font size of the text as a new feature.

III. Approach

A. Problem setting and formulation

In this work, we focus on the logical structure analysis of VSDs. The input is a document that contains a series of blocks extracted by the available layout analysis tool. Our goal is to extricate paragraphs from the document and recognize their relationships. To deal with this problem, we make a tree with each block as a node. To generate this tree, we identify the transition label between every two consecutive blocks that describe their relationships in this tree. As in [1], we also use five transition labels tran$_i$ between b_i (here before the i-th block) and b_{i+1} such as:

- *continuous*: b_i and b_{i+1} are continuous if they are both in a paragraph (Fig. 1(6))
- *consecutive*: b_i and b_{i+1} are consecutive if they are in two paragraphs at the same level (Fig. 1(7))
- *down*: b_{i+1} start a new paragraph its level is lower than the level of the paragraph that b_i belongs to (Fig. 1(6))
- *up*: b_{i+1} start a new paragraph its level is higher than the level of the paragraph that b_i belongs to (Fig. 1(8))
- *omitted*: b_1 is debris and omitted. (Fig. 1(9)) Now $trans_{i-1}$ is the relationship between b_{i-1} and b_{i+1}

While *down* here is well-defined, *up* is not clear and must be considered what level we should reach. To deal with this, a pointer is used for each up block, which b_j is denoted as a level that b_i belongs to ($pnt_i = b_j$ where $j < i$) (The example of detailed implementation is found in [1])

B. Logical structure analysis system

In this work, our logical structure analysis system is built based on a machine learning classifier and several handcrafted features. A machine learning model is more suitable for our research than a deep learning model because we can include textual, visual, and semantic cues in the model and it also needs less training data than a deep learning one.

To consider each block, our system extracts features from the paragraph with a group of four blocks and applies multi-class classification over five transition labels. Because *omitted* makes the target of transition changed, we delete *omitted* blocks in features extraction. With non-omitted transition, we extract features from $[b_{i-1}; b_i; b_{i+1}; bi + 2]$. Because we need to know the appearance of omitted blocks, we identify them by running the first prediction and using this information to identify other labels.

Our system can be changed flexibly to suitable several types of documents. In order to do that, we need to modify the document's features, so we build a list of features for each type by having a visual inspection of the training dataset (Table I). Some features are explained in detail in the study [1]. Since there are some different characters between TXT files and PDFs so we consider space characters as horizontal spacing and blank lines as vertical spacing. Therefore we can apply a system for TXT files like PDFs.

C. Pointer recognition system

We use a machine learning classifier to implement the pointer recognition system with handcrafted features. Because a down label is called, it creates a new level of the block so we need to point to this level when the up label is called. We extract all pair $[b_j, b_i]$ that $trans_j = down$ and $trans_i = up$, then we use features extracted from those pair to train a classifier to predict the pointer $(pnt_i = b_j)$

When the pointer at the block with down label (b_j), some features of the beginning block in the paragraph that contain b_j (we denote this as $b_{\text{begin}(j)}$) are so important. Therefore, we use $b_{\text{begin}(j)}$ to extract features from the pair $[b_j, b_i]$ such as:

- Consecutive numbering: Consider a number in b_i is contiguous to it in b_j and $b_{\text{begin}(j)}$ or not: Boolean features.
- Indentation: Consider the relationship about indentation of $[b_j, b_i]$ and $[b_{\text{begin}(j)}, b_{i+1}]$: Categorical features in [larger, smaller, stays the same].
- Left aligned: Consider $b_j, b_{i+1}, b_{\text{begin}(j)}$ are left aligned or not: Binary features.
- Transition counts: The number of blocks between b_i and b_j with down or with up, respectively: Numerical features.

We use those features for all document types despite the customizable pointer features. We used this strategy because of the successful implementation in those studies [1], [14].

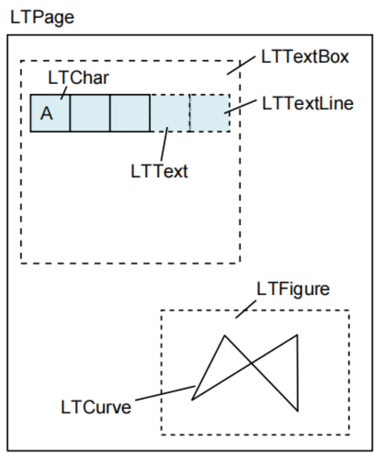
D. Fontsize Extraction

In this section, we detail our workaround and algorithm for extracting font size. We experimented with four types of visually structured documents (VSDs) in different file formats and languages.

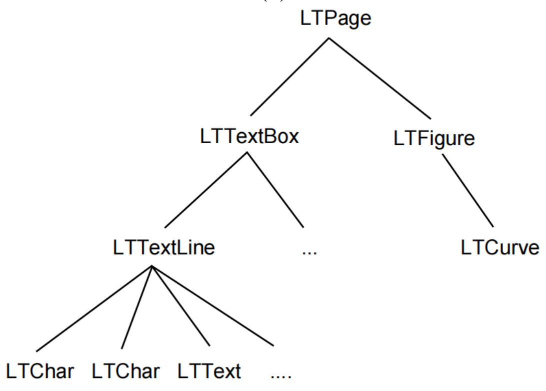
- $\textbf{Contract}^{\textbf{pdf}}_{\textbf{en}}$: English NDAs in PDF format.
- $\textbf{Law}^{\textbf{pdf}}_{\textbf{en}}$: English executive orders from local authorities.
- $\textbf{Contract}^{\textbf{txt}}_{\textbf{en}}$: English NDAs in visually structured plain text format.
- $\textbf{Contract}^{\textbf{pdf}}_{\textbf{ja}}$: Japanese NDAs in PDF format.

For PDFs, we use *PDFMiner*[1] and extract each *LTTextLine*, roughly corresponding to each line of text, as a block. We have merged multiple *LTTextLines* where the *LTTextLines* overlap vertically (Fig. 2). For plain text, we just need to use each non-blank line of plain text as a block.

[1]https://euske.github.io/pdfminer/



(a)



(b)

Fig. 2: Layout objects (a) and its tree structure (b)

The focus of this research was to suggest more information for each block extracted from the PDF file, which is the font size. So why font size? In VSDs, we can easily see that the title is often larger than the content within it. Therefore, with the desire to increase the accuracy of the transition labels, we add to each block information about its font size.

In the following, we will show how we extract font size from PDF files. First, we perform layout analysis for the PDF datasets. Parsing a PDF file is generally time and memory-consuming because a PDF file has such a large and complex structure. However, we do not use all parts for most PDF processing tasks, instead only a few needed parts are used. Therefore *PDFMiner* only parses the content when it is necessary, also known as the lazy parsing strategy. You need to use at least two classes: *PDFPaser* and *PDFDocument* to parse PDF files. These two objects are associated with each other. *PDFParser* fetches data from a file, and *PDFDocument* stores it. In addition to the two classes mentioned above, You will also need *PDFPageInterpreter* to process the page contents and *PDFPageAggregator* extract the deceive to page

TABLE I: List of features for feature extraction.

Relationship		Blocks	Document type		
			Contract$_{\mathrm{en}}^{\mathrm{pdf}}$/Law$_{\mathrm{en}}^{\mathrm{pdf}}$	Contract$_{\mathrm{en}}^{\mathrm{txt}}$	Contract$_{\mathrm{ja}}^{\mathrm{pdf}}$
Visual features					
V1	Indentation	1-2,2-3	v	v	v
V2	Indentation after erasing numbering	1-2,2-3		v	v
V3	Centered	2,3	v	v	v
V4	Line break before right margin	1,2	v	v	v
V5	Page change	1-2,2-3	v		
V6	Within top 15% of a page	2	v		v
V7	Within bottom 15% of a page	2	v		v
V8	Larger line spacing	1-2,2-3	v	v	v
V9	Justified with spaces in middle	2,3	v	v	v
V10	Similar text in a similar position	2	v		v
V11	Emphasis by spaces between characters	1,2			v
V12	Emphasis by parentheses	1,2			v
Textual features					
T1	Numbering transition	2	v	v	v
T2	Punctuated	1,2		v	v
T3	List start (/[-;:,]$/)	1,2	v	v	v
T4	List elements (/(;|, |and|or)$/)	2	v	v	
T5	Page number (strict)	1,2,3	v	v	v
T6	Page number (tolerant)	1,2,3	v	v	v
T7	Starts with "whereas"	3	v	v	
T8	Starts with "now, therefore"	3	v	v	
T9	Dictionary-like (includes ":" & not V4)	2,3	v	v	
T10	All capital	2,3	v	v	
T11	Contiguous blank field (underbars)	1-2,2-3	v	v	v
T12	Horizontal line ("*-=#%_+" only)	1,2,3		v	
T13	**Font-size (our feature)**	2	v		v
Semantic features					
S1	Language model coherence	1-2-3	v	v	v

The "Blocks" columns list blocks used to extract features for $trans_2$ (e.g. "1-2, 2-3" means $[b_{i-1}; b_i]$ and $[b_i; b_{i+1}]$ are used to extract two sets of features). Features with similar intended functionality are assigned the same feature name and implementations may vary for different document types.

aggregator to get LT object elements. *PDFResourceManager* is used to store shared resources such as fonts or images. A layout analyzer returns a *LTPage* object for each page in the PDF document. This object contains child objects within the page, forming a tree structure (Fig. 2). After we have performed the layout analysis, we use these *layouts* to extract the necessary information. We use **For** loop to traverse objects inside *layouts*. We set it as lt_obj. If lt_obj is an instance of *LTTextLine* then extract the text content. Continue the for loop to go up one more level in the structure tree (Fig. 2(b)). i.e. *character* traversal in lt_obj. **If** *character* is an instance of *LTChar*, then extract the font size. After extracting the font size, we check the condition that our text length is greater than 0 or not. **If** the text we extracted above has a length greater than zero, then return text and font size. Going back to the first branch condition, **else if** lt_obj is an instance of *LTTextBox* or *LTFigure*, we can not extract anything, so we use recursion to continue. At this point, the input is no longer the layout, the new input is lt_obj.

IV. Experiments and results

A. Evaluation metrics

In experiments, we use F-score for the task that identifies relationships between the pairs of blocks including (1) the same paragraph, (2) sibling, and (3) ancestor-descendant relationships, respectively.

The number of K-folds we use in the evaluation process are customized individually for each dataset. Specifically, for Contract$_{en}^{pdf}$ and Law$_{en}^{pdf}$, we used five-folds cross-validation. But, we used twelve for Contract$_{en}^{txt}$ and fifteen for Contract$_{ja}^{pdf}$.

B. Baselines

We compared our system against the following baselines:

a) Visual: This baseline is purely based on visual cues; i.e. indentation and line spacing. For each consecutive pair of blocks, this baseline outputs (1) *continuous* when indentation is unchanged and line spacing is normal, (2) *consecutive* when indent is unchanged change and line spacing is larger than normal, (3) *down* for larger indents and (4) *up* for smaller indents. On *up*, it points back to the closest block with the same indentation.

b) NB: This baseline presents a method for preprocessing unstructured documents in general to estimate document structure. The method consists of three algorithms and the recommendation follows a rule-based approach. The three algorithms are: (1) one is based on style information, such as bold font; (2) another is based on numbered objects, such as sections; and (3) the other is based on a document's Table of Contents, which summarizes the document's structure. We focus on algorithm 2 because our implementation is the same as the feature numbering transition (T13) (Table I) [4].

TABLE II: Results for evaluation on IE perspective.

Relationship			Contract$^{pdf}_{en}$				Law$^{pdf}_{en}$				Contract$^{txt}_{en}$				Contract$^{pdf}_{ja}$			
			Visual	NB	MTP	Ours	Visual	NB	MTP	Ours	Visual	NB	MTP	Ours	Visual	NB	MTP	Ours
Same paragraph	Micro	P	**0.982**	0.484	0.944	0.947	**0.891**	0.219	0.858	0.835	**0.993**	0.540	0.983	0.988	0.446	0.402	**0.973**	0.968
		R	0.683	0.947	0.951	**0.952**	0.681	**0.969**	0.957	0.954	0.708	0.917	**0.978**	0.971	0.552	**0.985**	0.966	0.967
		F	0.806	0.641	0.947	**0.948**	0.772	0.357	**0.905**	0.890	0.826	0.680	**0.980**	0.979	0.494	0.571	**0.969**	0.968
	Macro	P	**0.980**	0.644	0.955	0.956	0.906	0.328	**0.936**	0.944	**0.990**	0.595	0.969	0.976	0.481	0.478	**0.971**	0.959
		R	0.670	**0.966**	0.951	0.951	0.634	**0.974**	0.951	0.944	0.746	0.934	**0.976**	0.973	0.527	**0.985**	0.956	0.952
		F	0.782	0.736	0.948	**0.949**	0.731	0.452	**0.933**	0.929	0.847	0.687	0.971	**0.973**	0.450	0.617	**0.955**	0.947
Siblings	Micro	P	0.332	0.677	**0.841**	0.808	0.430	0.647	**0.849**	0.828	0.397	0.780	0.784	**0.849**	0.106	0.151	0.699	**0.770**
		R	0.323	0.765	0.736	**0.779**	0.283	0.504	0.712	**0.793**	0.481	**0.763**	0.723	0.725	0.506	0.571	0.691	**0.754**
		F	0.328	0.718	0.785	**0.793**	0.341	0.567	0.774	**0.810**	0.435	0.772	0.752	**0.782**	0.176	0.238	0.695	**0.762**
	Macro	P	0.443	0.678	**0.791**	0.779	0.598	0.493	**0.793**	0.797	0.482	0.677	**0.814**	0.803	0.347	0.237	0.719	**0.769**
		R	0.427	0.691	0.751	**0.781**	0.417	0.379	0.696	**0.720**	0.557	0.603	**0.758**	0.701	0.506	0.536	0.663	**0.740**
		F	0.337	0.650	0.748	**0.750**	0.410	0.385	0.724	**0.734**	0.435	0.605	**0.754**	0.729	0.292	0.283	0.671	**0.738**
Descendants	Micro	P	0.381	0.184	0.502	**0.596**	**0.717**	0.132	0.456	0.535	0.239	0.190	0.541	**0.664**	0.536	0.125	0.577	**0.788**
		R	0.123	**0.879**	0.807	0.831	0.303	**0.881**	0.858	0.855	0.048	**0.888**	0.771	0.836	0.340	0.580	**0.826**	0.811
		F	0.186	0.304	0.619	**0.694**	0.409	0.229	0.596	**0.658**	0.080	0.313	0.635	**0.740**	0.416	0.205	0.679	**0.799**
	Macro	P	0.295	0.242	0.655	**0.678**	0.438	0.173	0.581	**0.608**	0.193	0.269	0.639	**0.699**	0.462	0.122	0.737	**0.832**
		R	0.194	**0.848**	0.798	0.822	0.314	0.764	0.837	**0.855**	0.072	**0.859**	0.735	0.758	0.358	0.519	**0.834**	0.819
		F	0.203	0.340	0.641	**0.681**	0.327	0.230	0.617	**0.651**	0.096	0.367	0.625	**0.673**	0.372	0.195	0.739	**0.803**
Accuracy	Micro		0.772	0.778	0.914	**0.921**	0.827	0.685	0.908	**0.922**	0.587	0.674	0.828	**0.863**	0.618	0.623	0.940	**0.958**
	Macro		0.686	0.679	0.889	**0.891**	0.732	0.427	0.840	**0.853**	0.571	0.580	0.841	**0.852**	0.623	0.492	0.899	**0.916**
Average F1	Micro		0.440	0.555	0.784	**0.812**	0.507	0.384	0.758	**0.786**	0.447	0.588	0.789	**0.834**	0.362	0.338	0.781	**0.843**
	Macro		0.441	0.576	0.779	**0.793**	0.489	0.356	0.758	**0.771**	0.459	0.553	0.783	**0.792**	0.372	0.365	0.788	**0.829**

”Micro”: Micro-average, ”Macro”: Macro-average, ”P”: Precision, ”R”: Recall, ”F”: F1 score

c) MTP: [1] This baseline is a combination of multimedia markers. i.e. visual (such as indentation and line spacing), textual (such as section numbering and punctuation), and semantics (such as language model coherence) cues. The formula used here is a transition parser that predicts a transition label between each consecutive pair of text fragments.

C. Implementation Details

Our system makes use of a modular and customizable design and is implemented in Python. Users can put in force a brand new function extractor clearly through writing a new function extractor class wherein every function is applied as its class function. For example, $@single\_input\_feature([1])$ denotes that the following function should be carried out to the second block of every context. Likewise, the functions for pointer identity may be carried out with the aid of using marking a function with $@pointer\_feature()$, which takes a candidate block b_j ($tb1$), a goal block b_i ($tb2$), the block subsequent to the goal block b_{i+1} ($tb3$) and $b_{head(j)}$ ($head\_tb$) as an input. For each document, a feature extractor object is initialized. In which all features are automatically gathered to become a feature vector. To facilitate implementation, a new feature extractor can inherit from an existing feature extractor.

After establishing the feature that we propose, we conduct experiments on datasets where for each dataset we evaluate the contribution of features using random forest (Fig. 3). It can be observed that our feature (font size) is one of the features that has contributed a lot. In addition, we found that the *page_like2* and *mask_continuation* features made little or no contribution. Therefore, we have considered removing these features.

Finally, we used Random Forest and Decision Tree as the transition and pointer classifiers respectively. The settings for both models are described in Table III and IV.

TABLE III: Set of hyperparameters for each dataset for transition classifier

Document Type	RandomForestClassifier		
	n_estimators	min_samples_split	max_features
Contract$^{pdf}_{en}$	400	5	'log2'
Law$^{pdf}_{en}$	300	5	'sqrt'
Contract$^{txt}_{en}$	300	5	'sqrt'
Contract$^{pdf}_{ja}$	300	2	'sqrt'

TABLE IV: Set of hyperparameters for each dataset for pointer classifier

Document Type	DecisionTreeClassifier		
	max_leaf_nodes	criterion	random_state
Contract$^{txt}_{en}$	2	'entropy'	None
Contract$^{pdf}_{ja}$	2	'gini'	0
	RandomForestClassifier		
	n_estimators	min_samples_split	max_features
Contract$^{pdf}_{en}$	400	5	'log2'
Law$^{pdf}_{en}$	100	2	'sqrt'

D. Results and Discussion

1) F-score comparison: Structure evaluation is shown on Table II. Our system obtained micro-average structure prediction accuracy of 0.921 for Contract$^{pdf}_{en}$, 0.922 for Law$^{pdf}_{en}$, 0.863 for Contract$^{txt}_{en}$ and 0.958 for Contract$^{pdf}_{ja}$, significantly outperforming the best baselines with 0.914, 0.908, 0.828 and 0.940, respectively. Micro-average transition label prediction accuracies are 0.948 (Contract$^{pdf}_{en}$), 0.936 (Law$^{pdf}_{en}$), 0.959 (Contract$^{txt}_{en}$) and 0.929 (Contract$^{pdf}_{ja}$).

For *Average F1*, our model is superior to the baselines. Compared to MTP [1], our system shows larger gaps of 6.2% for Contract$^{pdf}_{ja}$, 4.5% for Contract$^{txt}_{en}$, 2.8% for both Contract$^{pdf}_{en}$ and Law$^{pdf}_{en}$. For relationships, our model also

TABLE V: Results for feature sets on IE perspective.

		Feature Set 1	Feature Set 2	Feature Set 3
Accuracy	Micro	**0.895**	0.892	0.860
	Macro	**0.862**	0.822	0.801
Average F1	Micro	**0.760**	0.728	0.676
	Macro	**0.761**	0.701	0.678

outweighs the baseline in some scenarios, for example, 6.7% and 18% in the Siblings and Descendants respectively. These experimental results verify the effectiveness of the proposed work by extracting and adding font size as a new feature.

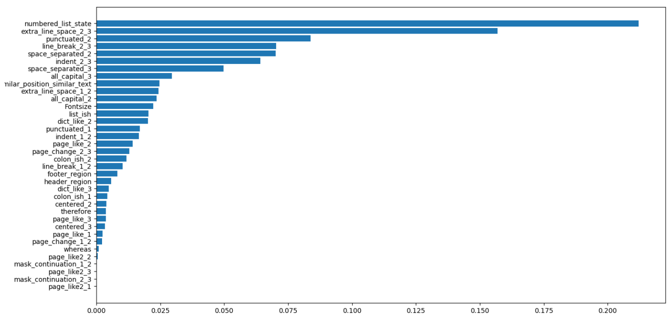


Fig. 3: Importance measurement of features (Contract$_{en}^{pdf}$)

2) Feature contribution: In this section, we go deeper into assessing the contribution of the remaining features. To do that, we evaluate the proposed model under three feature sets based on measuring feature importance (as shown in Fig. 3). In particular, we divide features into 3 subsets: Feature set 1 includes features from *numbered_list_state* to *footer_region* in Fig. 3; Feature set 2 includes *numbered_list_state* to *Fontsize*; Feature set 3 is the top 5 features with the highest contribution.

The experimental results are presented in Table V. As shown in the table, adding more features results in higher performance. This means that all selected features contribute to the performance of classification.

V. Conclusion

This paper introduces a new model to improve the logical analysis of visually structured documents. In order to do that, we introduce a new feature named font size that takes advantage of textual aspects to distinguish the structure of a VSD. The new feature based on the observation that the font size is an important indicator that can be used to represent the structure of a document. The new feature is combined with other visual features, textual features, and semantic features for training an analyzer. Experimental results on four legal datasets show that the new feature contributes to improving the performance of the model. The ablation study also shows the contribution of each feature.

Future work will investigate new features to improve the quality of the analysis. The model should be also tested on other genres and datasets.

Acknowledgement

We would like to thank Nguyen Hong Son, Huy-The Vu, and Minh-Tien Nguyen for their useful comments and supervision.

References

[1] Y. Koreeda and C. Manning, “Capturing logical structure of visually structured documents with multimodal transition parser,” in *Proceedings of the Natural Legal Language Processing Workshop 2021*. Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 144–154. [Online]. Available: https://aclanthology.org/2021.nllp-1.15

[2] V. W. Frederik Obermaier, Bastian Obermayer and W. Jaschensky, “About the panama papers,” in *Süddeutsche Zeitung*, 2016.

[3] M.-T. Nguyen, D. T. Le, and L. Le, “Transformers-based information extraction with limited data for domain-specific business documents,” *Engineering Applications of Artificial Intelligence*, vol. 97, p. 104100, 2021.

[4] Y. Hatsutori, K. Yoshikawa, and H. Imai, “Estimating legal document structure by considering style information and table of contents,” in *New Frontiers in Artificial Intelligence*, S. Kurahashi, Y. Ohta, S. Arai, K. Satoh, and D. Bekki, Eds. Cham: Springer International Publishing, 2017, pp. 270–283.

[5] C. G. Stahl, S. R. Young, D. Herrmannova, R. M. Patton, and J. C. Wells, “Deeppdf: A deep learning approach to extracting text from pdfs,” Oak Ridge National Lab.(ORNL), Oak Ridge, TN (United States), Tech. Rep., 2018.

[6] C. Soto and S. Yoo, “Visual detection with context for document layout analysis,” in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 3464–3470.

[7] Y. Xu, M. Li, L. Cui, S. Huang, F. Wei, and M. Zhou, “Layoutlm: Pre-training of text and layout for document image understanding,” in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 1192–1200.

[8] Y. Xu, Y. Xu, T. Lv, L. Cui, F. Wei, G. Wang, Y. Lu, D. A. F. Florêncio, C. Zhang, W. Che, M. Zhang, and L. Zhou, “Layoutlmv2: Multi-modal pre-training for visually-rich document understanding,” *CoRR*, vol. abs/2012.14740, 2020. [Online]. Available: https://arxiv.org/abs/2012.14740

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017.

[10] C. Sporleder and M. Lapata, “Automatic paragraph identification: A study across languages and domains,” in *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, 2004, pp. 72–79.

[11] C. Abreu, H. Cardoso, and E. Oliveira, “FinDSE@FinTOC-2019 shared task,” in *Proceedings of the Second Financial Narrative Processing Workshop (FNP 2019)*. Turku, Finland: Linköping University Electronic Press, Sep. 2019, pp. 69–73. [Online]. Available: https://aclanthology.org/W19-6410

[12] D. Ferrés, H. Saggion, F. Ronzano, and À. Bravo, “Pdfdigest: an adaptable layout-aware pdf-to-xml textual content extractor for scientific articles,” in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[13] M. Ostendorf, M. Collins, S. Narayanan, D. W. Oard, and L. Vanderwende, “Proceedings of human language technologies: The 2009 annual conference of the north american chapter of the association for computational linguistics,” in *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2009.

[14] S. Zhang, X. Ma, K. Duh, and B. V. Durme, “AMR parsing as sequence-to-graph transduction,” *CoRR*, vol. abs/1905.08704, 2019. [Online]. Available: http://arxiv.org/abs/1905.08704