法規・規格文書 PDF からの階層ツリー型データセット構築

平山 大世 [1]　松藤 彰宏 [2]　小川 祐輝 [2]　坂地 泰紀 [1*]　野田 五十樹 [1†]
[1] 北海道大学　[2] パナソニック株式会社
hirayama.h77@gmail.com　{matsufuji.akihiro, ogawa.yuki}@jp.panasonic.com
{sakaji, i.noda}@ist.hokudai.ac.jp

概要

法規・規格 PDF は多段にネストした階層構造と多様な番号体系を持つため，RAG 等で有用なツリー型データへの自動変換が困難である．本研究では，座標情報と境界条件を用いた見出し判定およびインデント分布に基づく階層構築により，PDF から階層ツリーを自動構築する手法を提案する．実際の法規文書を用いた評価では，適合率・再現率・F1 および親子関係の正解率でいずれも 1.00 を達成し，提案手法が文書構造を正確に抽出できることを示した．

1　はじめに

法規・規格文書とは，行政機関や標準化団体が定める規則・要件・試験方法などを記述した文書である．産業分野では適合性評価や設計要件の参照に用いられる．章・節・条・項といった階層構造を持ち，さらに「第〇条」「(1)」「ア」「(ア)」といった複数系列の箇条書きが多段にネストする（図 1）．こうした文書を検索拡張生成（RAG）等のアプリケーションで活用する場合，固定長チャンクによる単純な分割では，上位の文脈情報（所属する章・節）や項目間の参照関係が失われやすい．そのため，親子関係や兄弟順序を保持したツリー型データセットを自動構築できることが望ましい．

しかし，実運用においてこれらの文書は PDF 形式で配布されることが多く，以下の課題が伴う．

- **レイアウト由来の分断**：ページ境界により，論理的に連続する文が物理的に分断される．
- **図表由来のノイズ**：図表内のテキストや罫線文字が本文と混在して抽出される．
- **テキストレイヤの品質問題**：PDF のテキストレイヤに文字欠落や誤認識が存在する．
- **番号体系の多様性**：日本語法規文書特有の番号系列（イロハ順など）は，本文中の語彙（カタカナ語など）との衝突により誤検出しやすい．

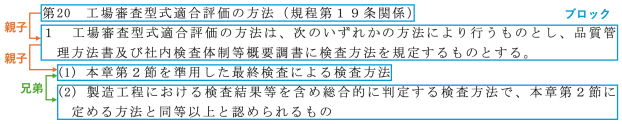

図 1　法規・規格文書におけるブロックと親子・兄弟関係．

本研究では，これらの課題を考慮した，PDF から階層ツリー型データセットを自動構築するパイプラインを提案する．貢献は以下の通りである．

1. pdfplumber による座標抽出から階層ツリー生成までを一貫処理するパイプラインを提案する．
2. 固定閾値に依存しない，境界条件と文書内分布に基づく汎用的な構造推定手法を示す．
3. 抽出誤りに対して保守的に対処し，構造の整合性を維持する設計を示す．
4. 評価用のアノテーションデータと評価指標を整備し，実文書における定量評価を行う．

2　関連研究

PDF からの情報抽出には pdfminer.six [1] や pdfplumber [2] が利用されるが，これらはテキストと座標の取得にとどまり，論理構造の復元は行わない．より高度な解析として，LayoutParser [3] や DocParser [4] など深層学習を用いたレイアウト検出が発展しているが，これらは視覚的ブロックの復元に主眼を置いており，日本語法規文書に特有の多系列番号における順序整合性は考慮していない．

文書の論理階層解析においては，学習ベースのモデル [5, 6, 7] が提案されている．しかし，これらの手法は十分な学習データを必要とし，多様な番号体系や表記揺れへの対応も保証されない．法規・契約文書に特化した研究 [8, 9] も進展しているが，本研究はルールベース処理と文書内分布推定により、法規特有の構造を復元する点で異なる．

* ORCID: 0000-0001-5030-625X
† ORCID: 0000-0003-1987-5336

また，RAG [10] の文脈保持手段として意味的チャンク化 [11, 12] が注目されているが，多くは Markdown 等の明示的な構造を持つテキストを前提としている．本研究は PDF から直接ツリー型データセットを構築する．なお，LLM による構造推定 [13, 14] は柔軟だが誤判定リスクがあるため，本研究では LLM を補助的なフィルタとして位置付ける．

3 提案手法

3.1 パイプラインの全体像

提案手法は，以下の処理パイプラインにより，入力 PDF から階層ツリー型データセットを生成する．処理は大きく 6 つの段階から構成される．

Step 1: PDF から座標付き単語を取得し，行単位にクラスタリング
Step 2: 行頭パターンから見出し型を抽出
Step 3: 行を論理ブロックに統合
Step 4: インデント分布から階層レベルを推定し，親子関係を決定
Step 5: 欠番検出，欠落項目の復元
Step 6: 祖先パスなどを求めて JSON 出力

3.2 出力データの形式

パイプラインはノード列を出力する．各ノードは以下の属性を持つ．

- **テキスト**：ノードの本文
- **見出し型**：章・節・項目などの分類（表 1）
- **親子関係**：親ノードと子ノードのリスト
- **兄弟関係**：前後の兄弟ノードへの参照
- **祖先パス**：ルートから当該ノードまでの列
- **番号マーカー**：箇条書きの番号
- **文脈テキスト**：上位ノードを結合した文脈情報

表 1 見出し型の定義と例．

見出し型	例	説明
章	第 1 章	文書の最上位区分
節	第 2 節	章の下位区分
大見出し	第 5 感知器…（第 8 条関係）	節の下位区分，条項への参照を含む
大項目	10 無線式感知器（第 16 号関係）	算用数字による項目
括弧数字	(4) 電界強度等の測定は…	括弧付き算用数字による項目
イロハ	ウ　プリント基板の材質は…	五十音順カタカナ（ア，イ，ウ…）による項目
括弧イロハ	（ア） 感知器は…	括弧付き五十音順カタカナによる下位項目
本文	（説明文）	見出しがない本文段落

なお、見出し型の事前定義は本手法の精度を左右する重要な要素である．

3.3 テキスト抽出

PDF からのテキスト抽出（Step 1）では，pdfplumber [2] を用いてページ上の単語と座標情報を取得する．抽出された単語は縦方向（y 座標）の近接性に基づいてクラスタリングし，行を構成する．同一行内の単語は横方向（x 座標）で整列し，読み順のテキストを得る．

図表領域からのノイズを抑制するため，表の罫線構造に基づくテーブル領域の除外を行うとともに，線分・矩形・曲線が密集する領域を図形密集領域として検出し除外する．密度の判定には四分位範囲に基づく外れ値検出を用いる．また，ページ番号などのノイズ行は正規表現パターンにより除去する．

3.4 見出し型判定

行頭のパターンマッチング（Step 2）により見出し型を判定し，対応する番号マーカーを抽出する．行頭の数字・記号が見出しマーカーか本文中の数値かは，直後に空白や句読点があるかどうかで判断する．

法規文書特有の五十音順カタカナ箇条書き（ア，イ，ウ…）は一般語彙と区別が困難である．本研究では語彙に依存せず，マーカー直後の文字境界に着目して誤検出を抑制する．具体的には，直後が空白・括弧・句読点などの区切りであれば見出しとし，直後にカタカナが連続する場合や「型/式/号/類/種/系/版/編」等が続く場合は，本文中の語の一部である可能性が高いため見出しと認識しない．

さらに，PDF テキストレイヤの空白欠落により，イロハマーカーと本文が連結して「ウプリント…」のように見出し検出が困難な場合がある．このような問題に対しては，本節では見出し候補の切り出しまでを行い，期待マーカーに基づく欠落検出と復元は後段のシーケンス検証と修復（3.7 節）で行う．

3.5 ブロック構築

テキスト行を論理的なブロック（Step 3）に統合する．ブロックは 1 つの見出しまたは本文段落に対応する単位である．基本原則は「見出し開始位置で新ブロックを開始する」であるが，PDF 特有の問題に対処するため以下の処理を行う．

ページ末尾で句点終端せずに途切れた本文は，次ページ先頭の行とインデント位置が整合する場合に

結合し，ページ跨ぎによる文の分断を修復する．同一行内に見出しと本文が混在する場合，「(第○条関係)」などの定型パターンを境界として分割し，分離された本文部分は独立した本文ブロックとして扱う．ブロック内部に次の番号マーカーが埋め込まれている場合（改行崩れなどが原因），先頭以外のマーカー出現を検出してブロックを分割することで，「見出しに本文が吸収される」「欠番が過剰に検出される」といった問題を抑制する．

また，括弧の全角・半角統一，見出しマーカー後の空白挿入，文中改行の除去などのテキスト正規化を行う．

3.6 ツリー構築

ブロック列から階層構造（Step 4）を推定する．ここでブロックをノードに変換する．まず，ブロックの行頭位置を収集・ソートし，隣接する座標間の差が許容幅を超える位置で区切ることで，離散的なインデントレベルに量子化する．許容幅は，座標間隔の中央値と中央絶対偏差の和として推定し，外れ値に頑健な閾値とする．

次に，見出し型から導出した階層レベル（表 1）に基づき，スタックを用いて親を決定する．処理対象ノードより深い（または同等の）レベルを持つノードをスタックから順次取り出し，残った最上位ノードを親として設定する．

3.7 シーケンス検証と修復

同一親配下の兄弟ノードについて，番号マーカーの順序整合性を検証する（Step 5）．「ア」「イ」の次が「エ」のように期待マーカーが欠落した場合，直前ノードのテキスト末尾に欠落マーカー（この例では「ウ」）で始まる段落が吸収されていないかを正規表現で検査する．発見時は段落境界で分割し，欠落マーカーを付与した新ノードを挿入することで，改行崩れに起因する構造誤りを修復する．

3.8 LLM フィルタ（補助的利用）

図表ノイズや罫線文字の除去が困難な場合，LLM によるノイズ判定を補助的に利用できる．本実験では GPT-OSS 20B [15] を使用した．ノードをチャンクに分割して LLM に入力し，各ノードについて保持または除去の判定を得る．ただし，見出し系・番号付きノードは削除対象から除外し，削除ノードの子は親に付け替えることで構造の破損を防ぐ．使用したプロンプトは付録 B に示す．

4 評価

4.1 データセット

評価対象として「火災報知設備の感知器及び発信機の検定細則」（総務省）の PDF 文書を使用した．この文書は，章・節・大見出し・括弧数字・イロハなど多様な見出し型を含み，3 階層以上のネストが頻出する．また，本文中に図表が含まれており，抽出時のノイズ処理が必要となる点でも提案手法の評価に適している．評価範囲は 11 ページ（第 4〜14 ページ）であり，正解ノード数は 189 である．正解データは人手で作成した．

4.2 評価指標

評価指標は，検出指標，属性指標，構造指標の 3 カテゴリに整理する．

4.2.1 正解との対応付け

正解・予測ノードの対応付けは，正規化テキストの文字列類似度[1)]に基づく貪欲法で行い，長文ノードから優先してマッチさせる．

4.2.2 検出指標

対応付けされたノード数を真陽性（TP），未対応の正解ノードを偽陰性（FN），未対応の予測ノードを偽陽性（FP）として，適合率・再現率・F1 を算出する．F1 は適合率と再現率の調和平均である．

4.2.3 属性指標

属性指標は，見出し型一致率（chapter, section, iroha 等の分類が正解と一致する割合），完全一致率（テキストが完全に一致する割合），正規化一致率（strict 正規化：NFKC，括弧・空白統一，lenient 正規化：大小文字・記号同一視，空白除去を適用した後の一致率），テキスト類似度（文字レベルでの抽出品質を測る平均類似度）で評価する．

4.2.4 構造指標

構造指標は，親カバレッジ（正解ノードの親が予測側にも対応付けられた割合），親子正解率（条件付き，親カバレッジを満たすノードについて親子関

1) Python 標準ライブラリ difflib の SequenceMatcher を使用．最長共通部分列（LCS）に基づく比率を返す．

係が正しい割合），兄弟順序正解率（同一親配下の兄弟ノードについて順序が正しいペアの割合）で評価する．

4.3 ベースライン

比較対象として，PDF から抽出した各行をそのまま 1 ノード 1 行としてルート直下に配置するベースラインを設定した．このベースラインは，構造解析を一切行わない最低限の抽出処理に相当し，提案手法の階層構造復元による改善効果を明確にする．全ノードがルート直下となるため，親子関係・兄弟順序のような構造指標は一致しない．

4.4 アブレーション

提案手法の全構成要素を有効化した設定と，主要構成要素を 1 つずつ無効化した設定でアブレーションスタディを実施した．無効化対象は以下に示す．

- 誤検出抑制（3.4 節）
- テキスト正規化（3.5 節）
- シーケンス検証・修復（3.7 節）
- LLM フィルタ（3.8 節）

4.5 結果

表 2 検出指標の比較．

指標	ベースライン	提案手法
適合率	0.31	1.00
再現率	0.76	1.00
F1	0.45	1.00

表 3 属性指標の比較．

指標	ベースライン	提案手法
見出し型一致率	0.05	1.00
完全一致率	0.02	0.80
正規化一致率（strict）	0.21	0.84
正規化一致率（lenient）	0.54	0.89
テキスト類似度	0.91	0.99

表 4 構造指標の比較．

指標	ベースライン	提案手法
親カバレッジ	0.97	1.00
親子正解率（条件付き）	0.00	1.00
兄弟順序正解率	0.00	1.00

提案手法は適合率・再現率・F1 および親子正解率・兄弟順序正解率でいずれも 1.00 を達成し，全ノードを正しく検出し階層構造を完全に復元した（表 2，表 4）．テキスト一致率は完全一致 0.80，strict 正規化 0.84，lenient 正規化 0.89，テキスト類似度 0.99 であった（表 3）．

アブレーションでは，テキスト正規化とシーケンス検証・修復の無効化で F1 がそれぞれ 0.98，0.97 に低下し，誤検出抑制無効化では見出し型一致率と親子正解率（条件付き）がそれぞれ 0.98，0.99 に低下し、F1 も 0.98 に低下した．LLM フィルタの無効化は主要指標に影響しなかった（付録 A）．

5 考察

提案手法が構造指標で完全一致を達成できた要因として，以下の 3 点が挙げられる．第一に，境界条件による見出し判定が，本文中の数値や一般語彙との誤検出を抑制した．第二に，文書内分布に基づくインデント量子化が，固定閾値では捉えられない文書固有のレイアウトに適応した．第三に，シーケンス検証による欠番検出と欠落項目の復元が，PDF 抽出時の改行崩れ（項目が複数行に分割される現象）に起因する構造誤りを修正した．

一方，テキスト一致率が完全一致に至らなかった主因は，PDF テキストレイヤの品質問題である．字形類似文字の誤認識（例：「1」と「l」，「基板」と「某板」）や図表キャプションの混入が確認された．strict 正規化と lenient 正規化の差分（0.84 → 0.89）は表記揺れに起因し，残りの 11%は PDF テキストレイヤの誤認識による実質的な抽出誤りである．

6 おわりに

本研究では，境界条件・文書内分布・シーケンス検証により固定閾値に依存しない構造推定を実現する，法規・規格 PDF 向け階層ツリー自動構築パイプラインを提案した．

法規文書（11 ページ，189 ノード）を対象とした評価では，構造の完全一致を達成した．

今後の課題として，以下を挙げる．

1. 複数文書での評価と汎化性能の検証
2. テキストレイヤ品質の定量分析と改善検討
3. 抽出した構造情報を活用した RAG システムでの有効性評価

また，本手法で補助的に用いている LLM フィルタについて，将来的には単なるノイズ除去だけでなく，条項の役割を付与する意味的アノテーションへの拡張も検討している．これにより，より充実した RAG 用データセットの構築が可能になるだろう．

参考文献

[1] Yusuke Shinyama and contributors. pdfminer.six. https://github.com/pdfminer/pdfminer.six. Accessed: 2025-12-14.

[2] Jeremy Svine. pdfplumber. https://github.com/jsvine/pdfplumber. Accessed: 2025-12-14.

[3] Zejiang Shen, Ruochen Zhang, et al. Layoutparser: A unified toolkit for deep learning based document image analysis. In **KDD**, 2021.

[4] Johannes Rausch, Victor Martinez, et al. Docparser: Hierarchical structure parsing of document images. In **ICDAR**, 2021.

[5] Yuta Koreeda and Christopher Manning. Contract document structure analysis based on visual and textual features. In **ACL**, 2021.

[6] Jianqiang Ma, et al. Hrdoc: Dataset and models for hierarchical document structure recovery. In **ACL**, 2023.

[7] Wei Li, et al. Dochienet: A large-scale dataset for document hierarchy parsing. In **AAAI**, 2024.

[8] Ilias Chalkidis, et al. Legal-bert: The muppets straight out of law school. In **EMNLP**, 2019.

[9] Manuel Real, et al. Structured parsing of legal documents with multimodal information. **Artificial Intelligence and Law**, 2024.

[10] Patrick Lewis, Ethan Perez, Aleksandra Piktus, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. In **NeurIPS**, 2020.

[11] Minghao Li, et al. Improving retrieval-augmented generation through semantic chunking. In **EMNLP**, 2023.

[12] Yunfan Gao, et al. A survey on retrieval-augmented generation. **ACM Computing Surveys**, 2024.

[13] Kai Zhou, et al. Docllm: Large language models for document understanding. In **ICCV**, 2023.

[14] Yuxuan Wang, et al. Structgpt: Large language models for structured document parsing. In **ACL**, 2023.

[15] OpenAI. Introducing gpt-oss. https://openai.com/index/introducing-gpt-oss/, 2025. Accessed: 2025-12-16.

A　アブレーション

提案手法の各構成要素を無効化した場合の結果を表 5 に示す．

表 5　アブレーションの結果．

設定	F1	見出し型一致率	親子正解率	完全一致率	strict 正規化	lenient 正規化	テキスト類似度
全構成	1.00	1.00	1.00	0.80	0.84	0.89	0.99
誤検出抑制を無効化	0.98	0.98	0.99	0.81	0.85	0.89	0.99
テキスト正規化を無効化	0.98	0.99	0.99	0.01	0.02	0.88	0.97
シーケンス検証・修復を無効化	0.97	0.99	0.99	0.80	0.82	0.87	0.98
LLM フィルタを無効化	1.00	1.00	1.00	0.80	0.84	0.89	0.99

テキスト正規化の影響　テキスト正規化を無効化すると，対応付け処理が影響を受け，F1 が 0.98 に低下した．また，strict 一致率が 0.02 と大幅に低下する一方，lenient 一致率は 0.88 と高く維持された．これは，正規化なしでは表記揺れ（空白・括弧・全角半角）が対応付けに影響する一方，実質的なテキスト差異は小さいことを示す．

誤検出抑制の影響　誤検出抑制を無効化すると，見出し型一致率が 0.98，親子正解率が 0.99 に低下し，F1 も 0.98 に低下した．これは，カタカナ語の先頭文字が箇条書きマーカーと誤認識されることや，空白欠落によりイロハマーカーが本文に連結することにより，一部のノードが未検出または誤分類されるためである．

シーケンス検証・修復の影響　シーケンス検証・修復を無効化すると F1 が 0.97 に低下した．これは，欠落したイロハ項目が直前ノード末尾に吸収される誤りや，連番規則から外れたマーカー（飛び番号・重複・誤抽出）が残り，一部ノードが未検出または誤った対応付けとなったためである．

その他の要素の影響　LLM フィルタおよび埋め込み分割を無効化しても本データセットでは主要指標に変化がなかった．これらの構成要素は，より複雑なレイアウトを持つ文書で有効性を発揮する可能性がある．

B　LLM フィルタのプロンプト

本研究で使用した LLM フィルタのプロンプト（システム指示）の主要部分を以下に示す．入力ノードごとに文脈テキストと正規化テキストを提示し，JSON 形式での回答を求めている．

```
あなたは法令 PDF から抽出したテキストブロックを分析するアシスタントです。
各ノードについて以下の 2 つを判定してください：

1. **keep/drop 判定**: 有意な本文なら 'keep'、ノイズなら 'drop'
2. **意味論的分類**: keep の場合のみ、以下から最も適切なタイプを選択
   選択肢: definition, requirement, test_method,
           exception, reference, heading, general

## 必ず drop すべきパターン
- 罫線記号の連続（─、━、|、┃、┣、┏、┓ など）
- 矢印や図形記号の混在（←、→、＼、／、□、○、◎など）
- 図表のラベル断片（「アンテナ」等が記号に埋もれている）
- 1〜2 語の断片的な日本語 + 多数の記号・罫線
- 図のキャプション単独（「図 1」「表 2」のみで説明がない）
- 意味不明な文字列（「ローー」「——' ＼ J」など）

## 意味論的タイプの説明
- definition: 用語の定義（「○○とは」など）
- requirement: 要件・規定（「○○であること」など）
- test_method: 試験方法（「○○試験を行う」など）

- exception: 例外規定（「ただし」「この限りでない」など）
- reference: 他条項への参照（「第○条による」など）
- heading: 見出しのみ（本文なし）
- general: 上記に当てはまらない一般的な説明

## 出力形式
JSON オブジェクトのみを出力:
{"node_id": {"decision": "keep",
          "semantic_type": "requirement"}, ...}
drop の場合:
{"node_id": {"decision": "drop",
             "semantic_type": null}}
```