

Acessilia at Dr.DocBench 2026: Agentic Document Parsing through Canonical Tree Adjudication

Marcelo Akira Inuzuka¹ Marcos Paulo Caetano Mendes Queiroz¹
Jhonata Fernandes Cordeiro² Eduardo Fraga Pereira¹

¹Instituto de Informática, Universidade Federal de Goiás (UFG)

²Centro Universitário Internacional (Uninter)

¹marceloakira@ufg.br

¹{marcospaulo2,eduardofraga}@discente.ufg.br ²fjhonata14@gmail.com

Acessilia Team — <https://github.com/A1lyDevs/acessilia>

Abstract

We present Acessilia, an agentic document-parsing architecture developed for the Dr.DocBench track of the DocInsights 2026 shared task. Independent structural agents use Docling and MinerU to generate alternative canonical document trees. A structural differ aligns these representations using textual and spatial evidence, exposing discrepancies for adjudication. The same mechanism can be recursively applied to specialised subtrees such as tables and mathematical expressions, while a deterministic serialiser converts the final canonical tree to the benchmark format. On the 509-page hidden test set, our final submitted system improves the official EvalAI score from 58.09 for the MinerU-only baseline to 70.37, reaching 74.67 in text, 72.30 in reading order, and 58.31 in TEDS; formula CDM remains near zero at 0.22. The results indicate that structural adjudication can substantially improve document parsing by reconciling complementary parser outputs.

1 Introduction

Document parsing aims to recover a structurally faithful machine-readable representation from page images containing text, tables, formulas, multi-column layouts, and other specialised objects. Benchmarks such as OmniDocBench (Ouyang et al., 2025) and Dr.DocBench (Yang et al., 2026) expose an important limitation of single-parser systems: performance depends on several heterogeneous components, including text recognition, reading order, table structure, and mathematical expressions. Different parsers therefore tend to exhibit complementary strengths rather than uniformly dominate across document types.

We present Acessilia, an agentic document-parsing architecture that exploits this complementarity through structural adjudication. Independent structural agents use Docling (Auer et al., 2024)

and MinerU (Wang et al., 2024b) to produce alternative interpretations of a page as canonical document trees. A Tree Differ aligns the resulting structures using textual and spatial evidence and identifies disputed nodes or subtrees. These discrepancies can then be resolved by an adjudicator that selects or edits the competing representations and may delegate specialised objects, such as tables or mathematical expressions, to dedicated processing paths. The resulting tree is finally serialised into the Dr.DocBench output format.

On the 509-page hidden test set, the final submitted configuration obtains an official EvalAI score of 70.37, compared with 58.09 for our MinerU-only baseline. The improvement is observed across text, reading order, and table structure, while formula CDM remains an unresolved limitation. Our main contributions are:

1. An agentic multi-parser architecture in which independent structural agents communicate through a versioned canonical document representation.
2. A structural adjudication mechanism that aligns alternative trees, exposes discrepancies, and supports hierarchical resolution at document and specialised-object levels.
3. An evaluation on the Dr.DocBench shared task showing a 12.28-point absolute improvement over our strongest single-parser baseline on the official test set.

Figure 1 summarises the agentic adjudication pipeline at a conceptual level; a more detailed technical architecture is presented in Section 4.

2 Related Work

Document parsing benchmarks. OmniDocBench (Ouyang et al., 2025) evaluates document parsing across heterogeneous layouts using component-specific metrics for text, tables,

Acessilia: Canonical Document Processing Pipeline

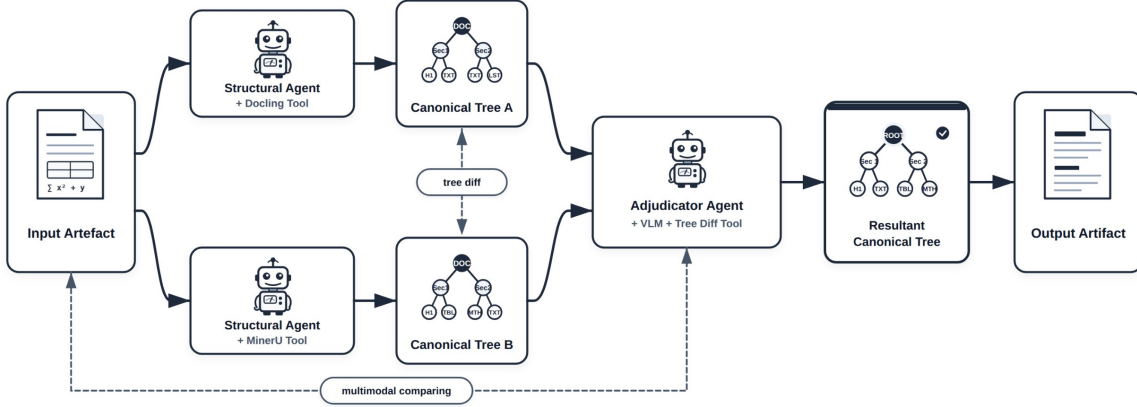


Figure 1: Conceptual overview of Acessilia. Independent structural agents use complementary document-processing tools to produce alternative canonical document trees. Their discrepancies are identified and adjudicated using structural and visual evidence, resulting in a final canonical representation that is serialised into the target document format.

formulas, and reading order. Dr.DocBench (Yang et al., 2026) extends this setting to difficult book pages and specialised content, including mathematical, chemical, and musical notation. These benchmarks highlight that document parsing is intrinsically multi-component: a system may perform well on text while failing on structural objects or reading order.

Generalist document parsers. Our system uses Docling (Auer et al., 2024) and MinerU (Wang et al., 2024b) as complementary parsing tools. Both combine layout analysis, OCR, table recognition, and mathematical-expression processing, but expose different structured representations and exhibit different failure modes. Other modular or end-to-end systems include PaddleOCR (Cui et al., 2025), Nougat (Blecher et al., 2023), GOT-OCR2.0 (Wei et al., 2024), olmOCR (Poznanski et al., 2025), and MonkeyOCR (Li et al., 2025). Acessilia operates above these recognisers, normalising their outputs into a shared canonical representation rather than introducing a new recognition model.

System combination and structural adjudication. Combining complementary recognisers has a long history in speech and OCR, including ROVER (Fiscus, 1997) and multiple-system OCR alignment (Lopresti and Zhou, 1997; Lund and Ringger, 2009; Wemhoener et al., 2013). For structured outputs, tree edit distance (Zhang and Shasha, 1989), structural change detection (Chawathe et al., 1996), and bipartite assignment (Kuhn, 1955) pro-

vide mechanisms for aligning alternative representations. Acessilia builds on these ideas by comparing canonical document trees and exposing discrepancies for adjudication at document and subtree levels, enabling the same structural mechanism to be applied to objects such as tables and mathematical expressions.

3 Task and Data

Dr.DocBench evaluates document parsing from page images into structurally faithful Markdown. The target representation preserves document structure through plain text, HTML tables, LaTeX mathematics, chemical notation, SMILES, MusicXML, and fenced code blocks. The public development split contains 986 pages from 66 documents, while the hidden test set contains 509 pages from 34 documents.

Evaluation combines four components: text edit distance, reading order, table TEDS (Zhong et al., 2020), and formula CDM (Wang et al., 2024a). The overall score averages the metrics applicable to each page and then averages across pages. We use the development split for system development and analysis, and report the official EvalAI result on the hidden test set as the primary evaluation of the final submitted system.

4 System Architecture

4.1 Overview

Acessilia follows an agentic architecture in which independent structural agents generate alternative interpretations of the same page and reconcile them through a shared canonical representation. As shown in Figure 2, two structural agents use Docling (Auer et al., 2024) and MinerU (Wang et al., 2024b) as tools to produce independent canonical document trees. A Tree Differ aligns these representations and exposes their discrepancies. An adjudicator agent then uses the source image and the competing structural hypotheses as evidence to select, edit, or delegate disputed subtrees. The resulting adjudicated tree is deterministically serialised to the Dr.DocBench output format.

4.2 Structural Agents and Toolbox

Document-processing tools are exposed through a capability-based REST Toolbox. A capability defines a versioned input/output contract under which different providers can be registered. The `document.structure.extract` capability receives a page image and returns structured content together with provenance information such as provider, version, and configuration.

The two structural agents invoke this same capability with different tools: Docling and MinerU. Docling combines layout analysis, OCR, table recognition, and formula processing, while MinerU provides corresponding layout, text, table, and mathematical-expression extraction. Treating these systems as tools rather than as the architecture itself allows downstream agents to operate independently of provider-specific formats.

4.3 Canonical Document Representation

Docling and MinerU expose different vocabularies, grouping strategies, and coordinate systems. Their outputs are therefore normalised into a versioned canonical document tree. Each block contains a stable identifier, semantic type, structured content, optional bounding box, and provenance metadata; reading order is represented by block order.

The canonical tree acts as the shared structured workspace between agents. Consequently, comparison and adjudication operate on a common representation rather than directly on parser-specific JSON or Markdown. Structured objects are represented as typed subtrees: tables preserve rows, cells, and spans, while mathematical objects can

retain a semantic representation before being converted to the notation required by the benchmark.

4.4 Tree Differ and Adjudication

Let T_D and T_M denote the canonical trees produced by the Docling and MinerU structural agents. The Tree Differ establishes correspondences between their blocks using textual and spatial evidence. Candidate pairs are matched one-to-one with the Hungarian algorithm (Kuhn, 1955). For blocks d and m , the matching cost is

$$c(d, m) = \lambda [1 - \text{IoU}(d, m)] + (1 - \lambda) [1 - \text{sim}(d, m)].$$

where IoU measures spatial overlap and sim is a normalised textual similarity. This follows the broader use of alignment and structural differencing for combining alternative recogniser outputs (Fiscus, 1997; Zhang and Shasha, 1989; Chawathe et al., 1996).

Matched nodes may agree in content and type or expose a discrepancy; unmatched nodes represent content found by only one agent. The Tree Differ therefore produces an explicit discrepancy set rather than immediately selecting one complete parser output.

For each disputed region, the adjudicator agent receives the competing subtrees together with the corresponding visual evidence from the original page. It can select one representation, edit the canonical structure, or delegate the region to a specialised processing path. Decisions are applied directly to the shared tree, producing a reconciled representation rather than simply choosing one parser output wholesale.

4.5 Hierarchical Specialist Adjudication

The same generate–compare–adjudicate pattern can be recursively applied to typed subtrees. For example, competing table interpretations can be compared as structured table trees, while mathematical expressions can be processed through a semantic mathematical representation. A specialist result is validated and inserted back into the document tree without changing the surrounding structure.

This hierarchical design allows generalist agents to recover the document as a whole while invoking specialised processing only where structural disagreement indicates that it is useful.

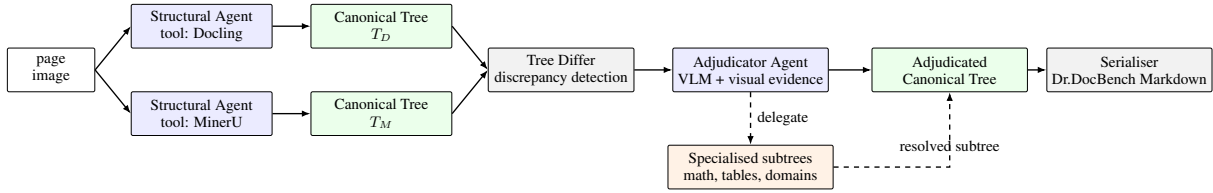


Figure 2: Technical architecture of ACESSILIA. Independent structural agents produce alternative canonical trees using Docling and MinerU. The Tree Differ detects structural discrepancies between the hypotheses, which are resolved by an adjudicator agent using visual and structured evidence. Disputed specialised objects can be delegated to domain-specific processing before being merged into the adjudicated canonical tree.

4.6 Serialisation and Validation

After adjudication, the final canonical tree is converted deterministically to the Dr.DocBench contract. Headings and paragraphs become Markdown text, tables are serialised as HTML with row and column spans, and mathematical objects are emitted as LaTeX. Other supported domain-specific representations can be mapped through the same serialisation layer.

Separating canonical representation from benchmark serialisation prevents output-format conventions from being embedded in the recognition agents. A final validation step checks page coverage and generates the official `predictions.jsonl` submission.

5 Experimental Setup

Data. We use the public Dr.DocBench development split, containing 986 pages from 66 documents, for system development and error analysis. Final performance is measured on the 509-page hidden test set through the official EvalAI evaluation. No additional training data or model fine-tuning is used.

System configuration. The evaluated pipeline uses Docling and MinerU as the two structural extraction tools (Auer et al., 2024; Wang et al., 2024b). Their outputs are converted into canonical document trees, aligned using textual and bounding-box information, and reconciled through structural adjudication. MinerU 3.4.5 uses the pipeline backend, while Docling is accessed through `docling-serve 1.32.0` with `Docling 2.127.0` (Auer et al., 2024). For visual adjudication over disputed regions, the adjudicator agent invokes Qwen2.5-VL-7B-Instruct (Bai et al., 2025) through a REST API service.

Evaluation. We use the official Dr.DocBench evaluator and report its four components: text accu-

racy, reading order, table TEDS, and formula CDM. The official EvalAI overall score on the hidden test set is our primary result. Development scores are used only for system construction and diagnosis and are not reported as estimates of hidden-test performance.

System context and code availability. ACESSILIA is fundamentally conceived as a production-oriented system for document content accessibility, currently in a pilot stage preceding full deployment. The experiments reported in this paper directly leverage ACESSILIA’s composable and reusable architecture—specifically its canonical document tree representation and capability-based REST Toolbox. To ensure transparency and full reproducibility, the core platform and the specific shared task evaluation pipeline are publicly accessible under the ACESSILIA Team GitHub organization: the Dr.DocBench experimental scripts hosted on the primary development branch¹, alongside the `accessilia-toolbox` repository² supplying the REST service infrastructure and parser provider wrappers.

6 Results

Table 1 reports the official EvalAI results on the 509-page hidden Dr.DocBench test set. The final ACESSILIA submission reaches an overall score of 70.37, compared with 58.09 for our MinerU-only baseline, an absolute gain of 12.28 points.

The largest improvement is observed in reading order, which increases from 53.44 to 72.30. Text improves by 7.02 points and TEDS by 10.95 points. These gains are consistent with ACESSILIA’s main design goal: reconciling complementary structural interpretations rather than relying on a single parser output.

¹<https://github.com/A11yDevs/accessilia/tree/develop/scripts/drbench>

²<https://github.com/A11yDevs/accessilia-toolbox>

System	Txt	RO	TEDS	CDM	Ovr.
MinerU	67.65	53.44	47.36	0.00	58.09
Acessilia	74.67	72.30	58.31	0.22	70.37

Table 1: Official EvalAI results on the 509-page hidden test set. Txt denotes text accuracy, RO reading order, and Ovr. the overall score.

Formula CDM remains near zero (0.22) and is the main unresolved limitation of the evaluated configuration. The overall improvement is therefore obtained almost entirely from gains in text, reading order, and table structure.

7 Analysis and Discussion

The strongest gain appears in reading order, which improves by 18.86 points. This result is particularly relevant to the proposed architecture because reading order depends on relations among blocks rather than on isolated text recognition. The improvement therefore suggests that reconciling alternative document structures can recover information that is difficult to obtain from a single parser output.

TEDS also improves substantially, from 47.36 to 58.31, while text accuracy increases from 67.65 to 74.67. Together, these results indicate that the benefit of adjudication is not restricted to one component: structural combination improves both textual content and higher-level document organisation. The canonical representation is central to this process because it allows parser-specific outputs to be compared and modified using the same block and subtree abstractions.

Formula CDM remains near zero (0.22) and is the main unresolved failure mode. This suggests that formula handling requires more than general structural adjudication, motivating the use of specialised mathematical subtrees and dedicated recognition or validation mechanisms. More generally, the results support the hierarchical design of Acessilia: generalist structural agents recover the page as a whole, while specialised processing can be reserved for object types or regions in which the competing representations remain insufficient.

We emphasize that the reported 12.28-point overall improvement reflects the performance of the full integrated pipeline, which combines Docling, MinerU, deterministic structural repairs, and tree adjudication. Because no full ablation study was performed to isolate each component, this performance gain cannot be attributed exclusively to vi-

sual adjudication or any single processing step, but rather to the synergy of the complete multi-parser architecture.

8 Limitations

The main limitation of the evaluated system is formula recognition: CDM reaches only 0.22 on the hidden test set, compared with 0.00 for the MinerU-only baseline. This indicates that the current pipeline does not yet produce mathematical representations that are reliably credited by the benchmark’s formula evaluator.

The adjudication pipeline also depends on the quality of the initial parser outputs and their spatial alignment. Errors shared by both structural agents may therefore remain undetected, while incorrect correspondences can affect downstream decisions. In addition, running multiple parsers and adjudication steps increases computational cost compared with a single-parser pipeline.

Our experimental evaluations with Qwen2.5-VL-7B-Instruct as the adjudicator agent revealed that visual adjudication provided relatively modest improvements compared to the primary gains driven by structural differencing and heuristic tree alignment. While the agentic pipeline successfully identified and isolated discrepant regions, the 7B-parameter vision-language model exhibited limited fine-grained discernment on complex, low-contrast, or densely packed document crops. This finding indicates that lightweight VLMs may lack the spatial and structural reasoning required for nuanced document adjudication. Future work will focus on integrating larger-capacity vision-language models (e.g., Qwen2.5-VL-72B or advanced frontier models) to improve visual discrimination during subtree reconciliation.

Finally, the reported results evaluate pages independently. Cross-page structure and some specialised object types are not yet fully exploited in the current configuration. These limitations motivate stronger mathematical specialists, improved structural validation, and document-level reasoning in future work.

9 Conclusion

We presented Acessilia, an agentic document-parsing architecture that combines complementary structural interpretations through canonical document trees, structural differencing, and adjudica-

tion. Rather than introducing a new recogniser, the system operates above existing parsers and reconciles their outputs in a shared structured representation that can also support hierarchical processing of specialised subtrees. On the 509-page hidden Dr.DocBench test set, the final submitted configuration improves the official EvalAI score from 58.09 for our MinerU-only baseline to 70.37. The strongest gains occur in reading order, text, and table structure, while formula CDM remains unresolved. These results show that structural adjudication is a practical approach for improving heterogeneous document parsing and provide a foundation for stronger mathematical and domain-specific specialists.

Ethics Statement

We use only the Dr.DocBench data released for the shared task and publicly available open-source parsing tools. No additional personal data are collected and no model fine-tuning is performed. Because document-parsing errors may affect downstream accessibility and information extraction, ACESSILIA preserves provenance in its canonical representation to support traceability and inspection of adjudication decisions.

Acknowledgements

The authors acknowledge the Laboratório Multiusuário de Computação de Alto Desempenho (LAMCAD/UFG) and the Centro de Excelência em Inteligência Artificial (CEIA/UFG) at the Federal University of Goiás (UFG) for providing the computational infrastructure for this research. The authors also acknowledge the Advanced Knowledge Center for Immersive Technologies (AKCIT) for its institutional support and the Fundação de Amparo à Pesquisa do Estado de Goiás (FAPEG) for its support of research and innovation.

During the preparation of this manuscript, the authors used ChatGPT (OpenAI) to assist in drafting, structuring, and refining the text across select sections (including the Introduction, Related Work, and Appendices). All AI-assisted outputs were thoroughly reviewed, fact-checked, and edited by the authors, who accept full responsibility for the final content, accuracy, and citations of this paper.

References

Christoph Auer, Maksym Lysak, Ahmed Nassar, Michele Dolfi, Nikolaos Livathinos, Panos Vage-

nas, Cesar Berrospi Ramis, Matteo Omenetti, Fabian Lindlbauer, Kasper Dinkla, Lokesh Mishra, Yusik Kim, Shubham Gupta, Rafael Teixeira de Lima, Valery Weber, Lucas Morin, Ingmar Meijer, Viktor Kuropiatnyk, and Peter W. J. Staar. 2024. [Docling technical report](#). *Preprint*, arXiv:2408.09869.

Shuai Bai et al. 2025. [Qwen2.5-vl technical report](#). *Preprint*, arXiv:2502.13923.

Lukas Blecher, Guillem Cucurull, Thomas Scialom, and Robert Stojnic. 2023. [Nougat: Neural optical understanding for academic documents](#). *Preprint*, arXiv:2308.13418.

Sudarshan S. Chawathe, Anand Rajaraman, Hector Garcia-Molina, and Jennifer Widom. 1996. Change detection in hierarchically structured information. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pages 493–504.

Cheng Cui, Ting Sun, Manhui Lin, Tingquan Gao, Yubo Zhang, Jiaxuan Liu, Xueqing Wang, Zelun Zhang, Changda Zhou, Hongen Liu, Yue Zhang, Wenyu Lv, Kui Huang, Yichao Zhang, Jing Zhang, Jun Zhang, Yi Liu, Dianhai Yu, and Yanjun Ma. 2025. [PaddleOCR 3.0 technical report](#). *Preprint*, arXiv:2507.05595.

Jonathan G. Fiscus. 1997. A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (ROVER). In *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 347–354.

Harold W. Kuhn. 1955. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1–2):83–97.

Zhang Li, Yuliang Liu, Qiang Liu, Zhiyin Ma, Ziyang Zhang, Shuo Zhang, Zidun Guo, Jiarui Zhang, Xinyu Wang, and Xiang Bai. 2025. [MonkeyOCR: Document parsing with a structure-recognition-relation triplet paradigm](#). *Preprint*, arXiv:2506.05218.

Daniel Lopresti and Jiangying Zhou. 1997. Using consensus sequence voting to correct OCR errors. *Computer Vision and Image Understanding*, 67(1):39–47.

William B. Lund and Eric K. Ringger. 2009. Improving optical character recognition through efficient multiple system alignment. In *Proceedings of the 9th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL)*.

Linke Ouyang, Yuan Qu, Hongbin Zhou, Jiawei Zhu, Rui Zhang, Qunshu Lin, Bin Wang, Zhiyuan Zhao, Man Jiang, Xiaomeng Zhao, Jin Shi, Fan Wu, Pei Chu, Minghao Liu, Zhenxiang Li, Chao Xu, Bo Zhang, Botian Shi, Zhongying Tu, and Conghui He. 2025. OmniDocBench: Benchmarking diverse PDF document parsing with comprehensive annotations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Jake Poznanski, Jon Borchardt, Jason Dunkelberger, Regan Huff, Daniel Lin, Aman Rangapur, Christopher Wilhelm, Kyle Lo, and Luca Soldaini. 2025. [olmOCR: Unlocking trillions of tokens in PDFs with vision language models](#). *Preprint*, arXiv:2502.18443.

Bin Wang, Fan Wu, Linke Ouyang, Zhuangcheng Gu, Rui Zhang, Renqiu Xia, Bo Zhang, and Conghui He. 2024a. [Image over text: Transforming formula recognition evaluation with character detection matching](#). *Preprint*, arXiv:2409.03643.

Bin Wang, Chao Xu, Xiaomeng Zhao, Linke Ouyang, Fan Wu, Zhiyuan Zhao, Rui Xu, Kaiwen Liu, Yuan Qu, Fukai Shang, Bo Zhang, Liqun Wei, Zhihao Sui, Wei Li, Botian Shi, Yu Qiao, Dahua Lin, and Conghui He. 2024b. [MinerU: An open-source solution for precise document content extraction](#). *Preprint*, arXiv:2409.18839.

Haoran Wei, Chenglong Liu, Jinyue Chen, Jia Wang, Lingyu Kong, Yanming Xu, Zheng Ge, Liang Zhao, Jianjian Sun, Yuang Peng, Chunrui Han, and Xiangyu Zhang. 2024. [General OCR theory: Towards OCR-2.0 via a unified end-to-end model](#). *Preprint*, arXiv:2409.01704.

David Wemhoener, Ismet Zeki Yalniz, and R. Manmatha. 2013. Creating an improved OCR service by merging multiple OCR outputs. In *Proceedings of the 12th International Conference on Document Analysis and Recognition (ICDAR)*.

Minglai Yang, Xinyan Velocity Yu, Pengyuan Li, et al. 2026. [Dr. DocBench: A comprehensive benchmark for expert-level and difficult document parsing](#). <https://github.com/2077AI/DrDocBench>. *Preprint*, arXiv:2606.01393.

Kaizhong Zhang and Dennis Shasha. 1989. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing*, 18(6):1245–1262.

Xu Zhong, Elaheh ShafieiBavani, and Antonio Jimeno Ypes. 2020. Image-based table recognition: Data, model, and evaluation. In *Computer Vision – ECCV 2020*, pages 564–580. Springer.

A Canonical Document Representation

Acessilia uses a versioned canonical document representation as the shared structured state between structural extraction, comparison, adjudication, and serialisation. Its purpose is to isolate downstream components from provider-specific representations: Docling and MinerU may expose different schemas, labels, grouping strategies, and coordinate conventions, but both are converted to the same canonical structure before comparison.

A.1 Document and Block Structure

A canonical document contains document-level metadata, source information, and an ordered collection of sections and blocks. Block order represents reading order. Each block has a stable identifier and a semantic type drawn from a controlled vocabulary such as heading, paragraph, list, table, math, code, and image.

Blocks may additionally contain a source location, including page number and bounding box, and provider-specific provenance metadata. Spatial information is preserved because it is used by the Tree Differ when aligning alternative document interpretations.

A simplified canonical page is shown below:

```
{
  "schema_version": "1.0.0",
  "id": "doc-001",
  "source": {
    "name": "page_032.jpg",
    "type": ".jpg"
  },
  "metadata": {
    "provider": "mineru",
    "page": 32
  },
  "sections": [
    {
      "id": "sec-1",
      "title": "Introduction",
      "blocks": [
        {
          "id": "b-1",
          "type": "heading",
          "level": 1,
          "text": "Introduction",
          "source_location": {
            "page": 32,
            "bbox": [72, 80, 300, 104]
          }
        },
        {
          "id": "b-2",
          "type": "paragraph",
          "text": "..."
        },
        {
          "id": "b-3",
          "type": "table",
          "table_ast": {
            "body": [
```

```

    {
      "cells": [
        {
          "text": "...",
          "rowspan": 2,
          "colspan": 1
        }
      ]
    }
  ],
},
{
  "id": "b-4",
  "type": "math",
  "text": "E = mc^2"
}
]
}
]
}

```

A.2 Typed Subtrees

Structured objects are represented internally as typed subtrees rather than being reduced immediately to benchmark Markdown. For example, tables preserve rows, cells, and span information in a table AST. This allows two table interpretations to be compared structurally before either is serialised to HTML.

Mathematical expressions may likewise be associated with a specialised semantic representation, such as MathML, while retaining the textual or LaTeX form required by a particular output format. The same principle can be extended to other domain objects: the canonical representation stores the object structure, while serialisation determines its external notation.

This separation is important because adjudication operates on the internal structure rather than on already-flattened Markdown strings.

A.3 Spatial Information

Whenever available, a block stores its location in the source page:

```

"source_location": {
  "page": 32,
  "bbox": [x0, y0, x1, y1]
}

```

Provider coordinates are normalised before comparison. The Tree Differ uses these locations to-

gether with textual similarity to establish candidate correspondences between blocks generated by different structural agents.

Preserving geometry is particularly important for multi-column layouts, headers and footers, tables, and other cases in which textual similarity alone is insufficient to determine structural correspondence.

A.4 Provenance and Adjudication

The canonical representation also provides the natural location for provenance associated with extraction and adjudication. At minimum, the system retains the provider and configuration that produced a representation. Adjudication metadata can additionally record which alternative was selected or whether a block was modified or replaced by specialised processing.

Conceptually, a block may therefore carry information of the following form:

```

"metadata": {
  "provider": "mineru",
  "decision": "selected",
  "adjudicated": true
}

```

The exact metadata fields depend on the processing stage and are not required by the Dr.DocBench output contract. They are retained internally to support traceability and inspection of the final canonical tree.

A.5 Representation and Output Separation

The canonical tree is deliberately independent of the Dr.DocBench Markdown format. Its structured content is converted to benchmark-specific syntax only by the serialisation layer. For example,

Canonical object	Dr.DocBench output
paragraph	plain Markdown text
heading	ATX heading
table AST	HTML <table>
mathematical object	LaTeX display math
code object	fenced code block

This separation allows the same canonical representation to be reused by comparison and adjudication components without embedding benchmark-specific formatting decisions into the structural agents.

B Tree Differ and Adjudication Details

This appendix provides additional details on the structural comparison and adjudication mechanism

used by ACESSILIA. The goal of the Tree Differ is not to decide which parser is globally better, but to expose local discrepancies between two canonical document trees so that they can be resolved explicitly.

B.1 Block Matching

Let T_D and T_M denote the canonical trees generated by the Docling and MinerU structural agents. Each block contains a semantic type, textual content, and, when available, a bounding box.

For each candidate pair of blocks $d \in T_D$ and $m \in T_M$, ACESSILIA computes a combined matching cost:

$$c(d, m) = \lambda [1 - \text{IoU}(d, m)] + (1 - \lambda) [1 - \text{sim}(d, m)],$$

where IoU measures spatial overlap and sim denotes normalised textual similarity. Candidate correspondences are selected with one-to-one bipartite matching using the Hungarian algorithm (Kuhn, 1955).

This formulation allows alignment to exploit both layout and content. Spatial evidence is particularly useful when similar text occurs in multiple regions, while textual evidence helps when parser bounding boxes differ slightly.

B.2 Discrepancy Types

After alignment, block pairs are classified according to their structural relationship. The main cases are:

- **Agreed:** both agents produce compatible block types and sufficiently similar content;
- **Content discrepancy:** the block type is compatible, but textual or structured content differs;
- **Type discrepancy:** the same region is represented using different semantic block types;
- **Unilateral block:** a block appears in only one canonical tree;
- **Structural discrepancy:** the same content is split, merged, or ordered differently between the two trees.

These discrepancy classes are represented explicitly so that downstream adjudication can operate only on uncertain regions instead of reprocessing the complete page.

B.3 Adjudication

For a disputed region, the adjudicator receives the competing canonical representations together with the corresponding crop or visual context from the original page. The adjudicator can perform one of three high-level actions:

1. select one of the existing hypotheses;
2. edit or reconstruct the canonical subtree;
3. delegate the disputed object to specialised processing.

The resulting decision is applied to the shared canonical tree and may be recorded in its provenance metadata. In this way, adjudication can produce a representation that is different from either original parser output.

B.4 Hierarchical Resolution

Adjudication is recursive. If a disputed document block contains a structured object such as a table or mathematical expression, the same comparison pattern can be applied inside that subtree:

hypotheses $\rightarrow$ alignment $\rightarrow$ discrepancies
 $\rightarrow$ adjudication.

This allows the document-level agents to recover the global structure while specialised agents operate only on objects that require additional resolution.

B.5 Processing Outline

Algorithm 1 summarises the process.

The Tree Differ is therefore responsible for exposing structural differences, while the adjudication layer is responsible for deciding how each difference should be resolved.

C Specialist Representations

ACESSILIA treats specialised document objects as typed subtrees of the canonical document representation. This allows the same generate–compare–adjudicate pattern used at document level to be applied to tables, mathematical expressions, and other structured objects.

Algorithm 1 Canonical tree comparison and adjudication

Require: page image I , canonical trees T_D, T_M

Ensure: adjudicated canonical tree T^*

```

1: extract blocks and spatial metadata from  $T_D$ 
   and  $T_M$ 
2: compute pairwise textual and spatial matching
   costs
3: align candidate blocks using Hungarian match-
   ing
4: classify matched and unmatched blocks as
   agreements or discrepancies
5: initialise  $T^*$  from the aligned canonical struc-
   ture
6: for each discrepancy  $r$  do
7:   collect competing subtrees and visual evi-
   dence from  $I$ 
8:   obtain an adjudication decision
9:   if  $r$  requires specialised processing then
10:    delegate  $r$  to the corresponding specialist
11:   end if
12:   update  $T^*$  with the resolved representation
13: end for
14: return  $T^*$ 

```

C.1 Tables

Tables are represented internally as structured table trees rather than as plain text. The canonical representation preserves rows, cells, and span information, allowing alternative table interpretations to be compared before serialisation.

A simplified table subtree has the form:

```

{
  "type": "table",
  "table_ast": {
    "body": [
      {
        "cells": [
          {
            "text": "...",
            "rowspan": 1,
            "colspan": 2
          }
        ]
      }
    ]
  }
}

```

After adjudication, the table tree is serialised to HTML with rowspan and colspan attributes as

required by Dr.DocBench.

C.2 Mathematical Expressions

Mathematical objects are treated as structured subtrees that can be processed independently from surrounding prose. The architecture supports associating such objects with a semantic mathematical representation, such as MathML, while retaining a LaTeX form for benchmark serialisation.

Conceptually, this separates:

$$\begin{aligned} \text{semantic math structure} &\rightarrow \text{adjudication} \\ &\rightarrow \text{LaTeX output.} \end{aligned}$$

This separation allows mathematical specialists to operate on a representation better suited to structural comparison without coupling the internal model to the output syntax expected by Dr.DocBench.

C.3 Other Domain Objects

The same mechanism can be extended to other specialised object types present in difficult documents. Examples include chemical formulas and structures, reaction schemes, music notation, and code blocks. Their canonical representations may use domain-specific formats, while the outer document tree stores them as typed objects with provenance and source location.

Typical output mappings include:

Object type	External representation
Table	HTML
Mathematics	LaTeX
Chemical formula	$\text{\ce{\dots}}$
Chemical structure	SMILES
Music notation	MusicXML
Code	fenced code block

These specialised paths are invoked only when needed, preserving the general document structure while allowing object-specific processing at subtree level.

D Serialization Rules

The canonical document tree is independent of the Dr.DocBench output syntax. A deterministic serialisation layer converts each canonical block into the corresponding benchmark representation.

Canonical block	Dr.DocBench output
heading	ATX Markdown heading
paragraph	plain Markdown text
list	Markdown list items
table	HTML <table>
math	display LaTeX
code	fenced code block
image	omitted
page header/footer	omitted

For tables, the serialiser reconstructs HTML from the canonical table AST, preserving row and column spans. Mathematical objects are emitted as display LaTeX using the notation required by the benchmark. Code blocks preserve their language tag when available.

Separating serialisation from recognition and adjudication has two practical advantages. First, parser-specific output conventions do not propagate into the rest of the system. Second, benchmark-specific formatting can be changed without modifying the structural agents or the canonical representation.

Before submission, the generated outputs are validated for page coverage and format consistency. The submission builder then writes one prediction per evaluation page into the official predictions.jsonl format.

E Adjudicator Prompt and Decision Schema

The adjudicator agent processes only regions identified as discrepant by the Tree Differ. In our evaluated setup, Qwen2.5-VL-7B-Instruct serves as the adjudicator backend via REST API calls. Its input includes the corresponding page crop, the two competing canonical subtrees, and their local structural context.

A simplified prompt template is:

You are a document-structure adjudicator.

Inputs:

1. source image crop;
2. candidate subtree A;
3. candidate subtree B;
4. parent and neighboring blocks.

Choose one action:

- select_a
- select_b
- edit

- delegate

For select_a or select_b:
return the selected subtree.

For edit:
return a corrected canonical subtree.

For delegate:
return the required specialist type.

Do not invent unsupported content.
Preserve reading order and structure.
Return only valid JSON.

The decision uses the following compact schema:

```
{
  "action": "select_a",
  "confidence": 0.95,
  "specialist": null,
  "subtree": null
}
```

The action field may also contain select_b, edit, or delegate. For edit, the returned subtree contains the corrected representation. For delegate, the specialist field identifies a processing path such as table or math.

This design constrains the VLM to adjudicating structured alternatives instead of performing unrestricted page transcription. The same mechanism can therefore be applied recursively to document blocks and specialised subtrees.

F Additional Experimental Details

This appendix records additional details from system development that are useful for reproducibility but are not required to understand the final architecture.

F.1 Development Data

The public Dr.DocBench development split contains 986 pages from 66 documents. During early development, we also used a stratified 120-page subset for rapid diagnosis and regression testing. This subset contains 50 documents and 33 subjects and deliberately over-samples pages with tables, isolated equations, multi-column layouts, and other difficult structures.

The 120-page subset was used only as a diagnostic set. Final system development and validation

were performed on the full development split, while the hidden 509-page test set was used only through official EvalAI submissions.

F.2 Parser Configuration

The final system uses Docling and MinerU as the two structural extraction tools. MinerU 3.4.5 runs with the pipeline backend. Docling is accessed through docling-serve 1.32.0 with Docling 2.127.0.

Both parsers process the same page image independently. Their outputs are persisted with block text, semantic type, bounding boxes, page dimensions, and provider provenance before canonicalisation and structural comparison.

F.3 Alignment Parameters

The Tree Differ combines textual and spatial evidence during block matching. For the final configuration, the matching cost uses equal weighting between textual similarity and bounding-box overlap: $\lambda = 0.5$.

Since $c(d, m)$ represents a matching cost (where lower values indicate closer structural alignment), candidate pairs are accepted as aligned blocks only if their cost satisfies $c(d, m) \leq \tau$, with the acceptance cost threshold set to $\tau = 0.6$. Candidate correspondences with a cost exceeding $\tau = 0.6$ are classified as unaligned discrepancies. One-to-one assignments are obtained with the Hungarian algorithm before discrepancy classification.

F.4 Preprocessing and Structural Repair

Development analysis identified several recurring structural failure modes, including fragmented paragraphs, duplicated text, misplaced headers and footers, page numbers, overlapping regions, and rotated pages. The final pipeline includes deterministic preprocessing and structural-repair rules for these cases before final serialisation.

Pages detected with incorrect orientation are normalised before structural extraction. Repair rules operate on canonical blocks rather than directly on the final Markdown, preserving the separation between recognition, adjudication, and serialisation.

F.5 Evaluation Protocol

The official Dr.DocBench evaluator is used for the reported test-set results. It computes text, reading order, table TEDS, and formula CDM components and aggregates the applicable metrics at page level.

Development evaluation is used for diagnosis and system construction only. Because the develop-

ment split was repeatedly inspected during implementation, its scores should not be interpreted as independent estimates of hidden-test performance.

G Infrastructure and Reproducibility

Development and evaluation runs were executed in a cluster environment using one Slurm job per parser run. Each pipeline job allocated 32 CPU cores and 128 GB of RAM on a DGX H100 node; no GPU resources were directly requested for the main Slurm processing jobs.

This CPU-only allocation was possible because the heavy vision-language model (Qwen2.5-VL-7B-Instruct) was hosted on a dedicated, separate GPU inference endpoint. The main ACESSILIA pipeline communicated with the VLM remotely via HTTP/REST API calls during the adjudication phase. Local parsing tools (docling-serve and mineru-api) and the ACESSILIA REST Toolbox were executed locally within the job environment on the CPU allocation. The page-processing loop was re-entrant: previously produced outputs were cached to allow interrupted jobs to resume without recomputation.

Model weights and package caches were stored on node-local storage to reduce startup overhead. Provider outputs, canonical representations, and intermediate structural metadata were persisted so that comparison, adjudication, and serialisation could be rerun independently of the original recognition step.

The implementation records provider name, version, and configuration as provenance in the canonical representation. This allows submitted results to be associated with the exact parser configuration used to generate each structural hypothesis.