DOCUMENT STRUCTURE BEYOND HEADING HIERARCHY: DATASET, METHOD, AND BENCHMARK

Anonymous authors
Paper under double-blind review

ABSTRACT

Documents organize meaning through their structure. Explicit structure helps retrieval and question answering, yet document parsing optimizes word-level accuracy and page-level fidelity while structure recovery targets the heading hierarchy alone. That formulation under-determines the structure of real documents: a parent need not be a title, an element may fall outside the ordered forest and rejoin it through a reference link to a target many pages away, and a unit need not be what one page yields, since page breaks split fragments and running headers take no part in it. We reformulate the problem as the joint prediction of a structure $G = (V, E_h, E_r)$ over a document: semantic elements that persist across page boundaries and carry a flag for taking part, a forest whose internal nodes need not be titles and whose groups may be named by virtual nodes, and directed reference edges from an anchor to a target anywhere in the document. We instantiate this with **DocVine**, a human-annotated dataset of 1,217 documents and 12,736 pages in which all three are annotated over the same elements. We design **VinePipe**, a staged pipeline, and train **VineVLM-2B**, a 2B vision language model fine-tuned to drive every stage of the pipeline. **VinePipe** with **VineVLM-2B** beats task-specific hierarchy parsers and frontier vision language models on DocVine, and leads on MPDocBench-Parse under its own protocol.

1 INTRODUCTION

The information in a document lives not only in its words but also in its structure. Representations that carry hierarchy or explicit structure consistently help retrieval and answer questions Sarthi et al. (2024); Saad-Falcon et al. (2024); Xu et al. (2026b;a). However, recent studies on document parsing have largely overlooked structural information, focusing instead on word-level accuracy and page-level fidelity.Cui et al. (2021); Smith (2007); Zhong et al. (2019); Jaume et al. (2019); Xu et al. (2020); Huang et al. (2022); Cui et al. (2025); Niu et al. (2025); Tencent Hunyuan Vision Team (2025); Wei et al. (2025); Cai et al. (2026); Lu et al. (2026); Ouyang et al. (2025); Poznanski et al. (2025). A smaller stream of work targeting structure recovery focuses mainly on recovering the heading hierarchy Wang et al. (2020); Rausch et al. (2021); Ma et al. (2023); Xing et al. (2024). We argue that the formulation is inadequate. It under-determines the structure in three aspects.

First, a heading hierarchy requires every parent to be a title, which real documents violate constantly. A paragraph may open with a bold lead phrase that governs the paragraphs beneath it, and the items of a list are subordinate to the sentence that introduces the list. Many coherent groups carry no heading at all, but they need a node created to host them. A parser built to recover a heading hierarchy has nothing to emit in that position. Second, the heading hierarchy takes the structure to be an ordered tree. Many elements nevertheless fall outside that tree. A footnote marked by a superscript, a floating table, or a floating figure does not fit the notion of a node in that tree. Such elements join the structure through reference links instead. These links tie an anchor to a target, which may sit many pages away. Prior work confines these links to scholarly citation parsing Lopez (2009) or to relation prediction over single pages Chen et al. (2025c). Third, a unit is not always what a single page yields. Fragments split by a page break may belong to one element. Running headers, page numbers, and repeated table heads are units, but they take no part in the structure.

This paper closes this gap by widening the definition rather than by simply proposing a new algorithm or strengthening a model. We formulate the problem as the prediction of a structure

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

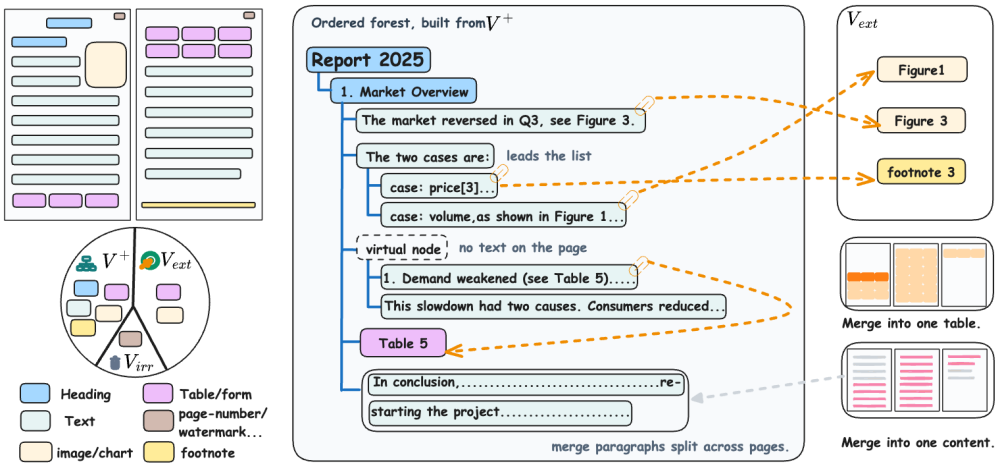


Figure 1: Predicted structure $G = (V, E_h, E_r)$. Units flagged irrelevant stay out of the structure. The hierarchy is an ordered forest whose internal nodes need not be titles and whose elements survive page breaks. Reference edges lie outside that forest. Each joins an anchor to a target that may sit pages away.

$G = (V, E_h, E_r)$ over a document. The node set V holds semantic elements that persist across page boundaries, each carrying a flag that records whether it takes part in the structure. The hierarchical edges E_h form a forest whose internal nodes need not be titles and whose groups may be named by virtual nodes. The reference edges E_r carry directed links from an anchor to a target anywhere in the document. The three components are defined jointly rather than assembled afterwards, so each can be measured on its own and all three can be measured together.

We instantiate this definition with *DocVine*, a corpus of 1,217 full documents totalling 12,736 pages. We split the documents into train, validation, and test sets. Realizing the definition takes work at two levels. The first is a pipeline. It recognizes the elements, builds the forest, and resolves the links in stages. We call the pipeline *VinePipe*. The second is a model. We fine-tune one small vision language model (2B) to drive every stage of the pipeline, and name it *VineVLM-2B*. Our approach outperforms task-specific hierarchy parsers and frontier vision language models on DocVine. It also leads on MPDocBench-Parse Zhou et al. (2026), an external benchmark that scores a narrower task, under that benchmark's own protocol. Case studies show what changes on documents where the widened structure decides the output.

Our contributions are summarized as follows:

- A widened definition of document structure understanding: predict one structure $G = (V, E_h, E_r)$ over a document. A heading hierarchy becomes a degenerate case of the problem.
- DocVine, a corpus and benchmark of 1,217 documents and 12,736 pages. It is the first to annotate the elements, the ordered forest, and the reference links of a document together.
- VinePipe, a pipeline that recognizes the elements, builds the ordered forest, and resolves the reference links, and VineVLM-2B, a vision language model fine-tuned to drive every stage of it.
- Experiments show that VinePipe with VineVLM-2B achieves the best results on DocVine, surpassing task-specific hierarchy parsers and prompted frontier vision language models. On MPDocBench-Parse, an external benchmark that scores a narrower task, the same approach leads under that benchmark's own protocol.

2 Related Work

Page content and layout parsing Document understanding was long identified with optical character recognition, which locates and transcribes text and measures character accuracy Smith (2007); Cui et al. (2021). Layout analysis grew beside it, from projection profiles and connected compo-

nents O'Gorman (1993); Nagy & Seth (1984) to large annotated corpora Zhong et al. (2019); Li et al. (2020); Jaume et al. (2019); Pfitzmann et al. (2022), with tables split off into their own track Smock et al. (2022) and reading order treated as a task of its own Wang et al. (2021). Models then moved from layout-aware language models Xu et al. (2020); Huang et al. (2022) to prompt-based generators Tang et al. (2023) and on to vision language parsers Cui et al. (2025); Niu et al. (2025); Zhang et al. (2025); Duan et al. (2026); Tencent Hunyuan Vision Team (2025); Wei et al. (2025); Cai et al. (2026); Lu et al. (2026). Page-level fidelity is scored by dedicated benchmarks Ouyang et al. (2025), and linearizing a PDF into flat text defines the other endpoint of the line Poznanski et al. (2025). What this line returns is a list of blocks in a reading order. Structure enters only at the margins, as a heading level read off the font or a table stitched across two adjacent pages Wang et al. (2026); Feng et al. (2026); Cui et al. (2026).

Hierarchy parsing A second line makes structure the target. Early work stays inside narrow settings such as form key–value pairs Wang et al. (2020), later formulations emit a hierarchy directly, weakly supervised from LaTeX source or generated end to end with corpora of their own Rausch et al. (2021; 2023), and line-level reconstruction turns the task into a benchmark of thousands of documents Ma et al. (2023); Xing et al. (2024). Decoding is what varies: a transition parser over a stack and a buffer Zhang et al. (2024), biaffine parent scoring Shin et al. (2026b), incremental construction chunk by chunk Li et al. (2026), and rule-filled language-model section trees Shin et al. (2026a). MinerU-PoPo goes furthest, yet its repair yields a list of integer heading levels handled separately for text and tables, and its links are confined to image references Xu et al. (2026a).

Reference links Within a page, DRGG predicts relation triplets over detected instances Chen et al. (2025c). Beyond a page, links appear only in narrow, mostly scholarly settings. GROBID only recovers scholar citation strings of one document Lopez (2009). LongDocBench comes closest to our third component, though its relations from tables and figures to their own captions, notes, and sources still attach an object to material that belongs to it, as do DocGenome's typed relations between scientific entities within a page Xia et al. (2024); Zou et al. (2026).

Structure-aware retrieval Multi-page retrieval systems still index flat text chunks and page images Cho et al. (2024); Chen et al. (2025a). Structure pays off there, and tree-organized indices improve long-document question answering Sarthi et al. (2024); Saad-Falcon et al. (2024); Xu et al. (2026b); Loison et al. (2026); Shin et al. (2026c).

3 Problem Definition

The problem of document structure understanding takes a document D as an ordered sequence of P page images $D = \{I_p\}_{p=1}^{P}$, and produces a structure

$$G = (V, E_h, E_r), \tag{1}$$

where V is the element set of G, E_h is the ordered forest over the elements that enter the hierarchy, and E_r is the set of reference links. Both edge sets are defined over V, and no noise node is an endpoint of either.

Elements. An element is one object of the document, and it takes one of two forms. A content element is a region of the visible content. A virtual element is one the structure requires and the hierarchy mounts, although the document prints nothing there. The visible content of the document partitions into content elements, so every region of content belongs to exactly one of them. An element carries bounding boxes $\mathrm{boxes}(v)$, one box for each region it occupies, text $\mathrm{text}(v)$, and a category $\mathrm{cat}(v)$. The box list, the text and the category of a virtual element are empty. A page break may cut one object apart, so a content element may occupy regions on several pages and a table continued across a page break is one content element.

The element set V holds every element of D. Each element carries one kind,

$$\mathrm{kind}(v) \in \{\text{in-tree},\ \text{out-tree},\ \text{irrelevant}\}, \tag{2}$$

and the kinds partition V,

$$V = V^{+} \cup V_{\mathrm{ext}} \cup V_{\mathrm{irr}}, \tag{3}$$

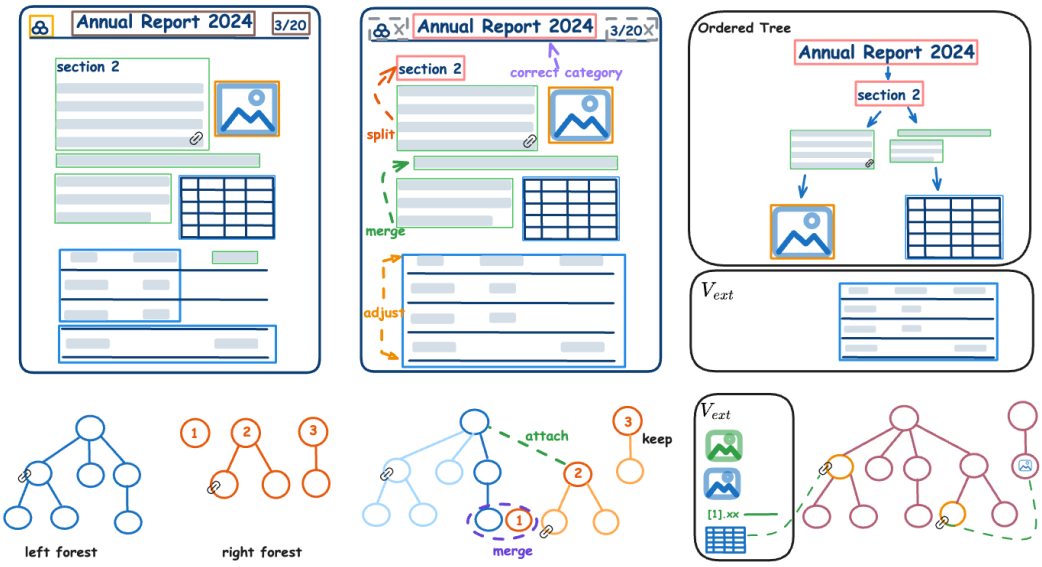


Figure 2: VinePipe. (1) Page regions are corrected and flagged as in-tree, out-of-tree, or irrelevant. (2) Per-page forest, with virtual nodes, are merged across pages into one forest, joining continued elements and attaching subtrees. (3) Explicit and implicit anchors are resolved to their targets.

with V^+ the in-tree elements, V_{ext} the out-of-tree elements, and V_{irr} the noise. An in-tree element is an element the hierarchy mounts, such as a heading, a paragraph, a list item, or a table. An out-of-tree element is material the document prints beside the hierarchy, such as a floating figure, a caption, or a bibliography entry, and the kind applies to the instance: the same figure is an in-tree element when the tree mounts it and an out-of-tree element when it stays afloat, and a reference link may point at either. A noise element is a running header, a footer, a page number, a restatement of a table head on a continuation page, or decoration.

Ordered forest. Hierarchical edges satisfy $E_h \subseteq V^+ \times V^+$, where $(u, v) \in E_h$ means that v is subordinate to u. The graph (V^+, E_h) is an ordered forest: each element has at most one parent, the graph has no cycles, and siblings under the same parent follow the reading order of the document. An internal element need not be a title. A paragraph may open with a bolded lead phrase that governs the paragraphs beneath it, and the items of a list are subordinate to the sentence that introduces the list. Such a parent is a content element of V^+ like any other, and where the document prints no such text, the parent is a virtual element. The elements of V^+ are exactly the elements of the tree.

Reference links. E_r is a set of directed links, $E_r \subseteq (V^+ \cup V_{\text{ext}}) \times (V^+ \cup V_{\text{ext}})$, so both endpoints are elements of the document, whether the hierarchy mounts them or leaves them beside it. Each link has one locatable anchor, which lies in a region of a content element, and several links may share one anchor. The target may be any element, a leaf, an internal element, or a virtual one, since a reference often names a group whose heading the document never prints. A link records an explicit reference to a figure, a table, a section, an appendix, or a citation. A link also records the relation between a superscript and its footnote, which is an out-of-tree element, and an implicit back-reference such as "as discussed in the previous chapter", whose target is a section or a chapter rather than a leaf. A link is a relation beside the structure. The target keeps its parent in the tree, or keeps its place beside the tree when it is out-of-tree, and `link_to` serializes the relation. Each link joins two distinct elements, and the pairs in E_r are distinct.

4 VinePipe: A Pipeline for Document Structure Understanding

We design VinePipe, a pipeline for document structure understanding, to predict $G = (V, E_h, E_r)$. Three stages run in sequence. The first turns page images into regions and gives every region its membership, which proposes the material that V is built from. The second assembles that material into the elements of V and settles E_h at the same time, since bringing pages together is what decides

that a table printed across two pages is one element rather than two, and the same stage creates the virtual elements that V also holds. The third settles E_r. What the stages fix therefore overlaps, and V is complete only once the tree is built.

4.1 Elements from pages

Detection. A coarse detector turns each page image I_p into page-level regions R_p, and every region carries a `category`, a bounding `bbox`, and recognized `text`. Two families of front-end return such candidates. The first locates them with a dedicated model Zhong et al. (2019); Li et al. (2020); Jaume et al. (2019); Pfitzmann et al. (2022); Smock et al. (2022); Xu et al. (2020); Huang et al. (2022); Shen et al. (2021); Lin et al. (2024). The second reads the page image with a vision language model and returns the regions together with their text Cui et al. (2025); Niu et al. (2025); Zhang et al. (2025). Either family serves behind the same candidate interface. A region is a candidate rather than an element: regions are the input of the pipeline and elements are its output, and the two are named apart throughout the paper.

Correction and Membership. A vision language model reads the page image together with the candidate regions and returns a corrected set of regions. Five actions carry the correction: *merge* joins adjacent regions, *split* breaks an over-merged region, *adjust* refines a boundary, *supplement* adds a missed region of visible text, and *recategorize* corrects a semantic `category`. A region whose boundary is modified has its text recognized again on the corrected region. The detector proposes and the model decides, so the coarse level only has to be plausible and the model settles the regions a page actually carries.

Every region receives exactly one membership, which places it in the in-tree stream, the out-of-tree stream, or the noise stream. Membership is a property of a region, and it determines the kind of the element that region becomes. The two content streams stay apart: no element is built from regions of both, so every element carries exactly one kind, and that kind is the membership its regions were given. The noise stream ends here. Its regions become the noise elements of V_{irr}, one element for each occurrence, and the membership is what keeps them out of the tree and out of every link.

4.2 Forest construction

One page at a time. Working from a single page, forest construction gives every in-tree region of that page a parent, and writes the records in reading order. A parent is the element that governs a group. Where the page prints no such text, the parent is a virtual element. The first page of a document usually yields one tree with a single root, because its topmost element is a heading that governs what follows. A later page rarely does. Its first element often governs nothing below it, since the section it belongs to began on an earlier page, and it may even be the tail of an element that a page break cut in two. The page then carries several top-level nodes instead of one, and the step keeps each of them as the root of its own tree. Which of those nodes sits under which parent is a question about the pages before it, so the cross-page step answers it and every top-level node is offered to the forest the preceding pages have built.

Across pages. Once every page carries a forest of its own, these forests are reduced to one. Merging page by page from the front is the obvious route, but it is slow: the whole process can only be sequentially applied. We instead reduce pairwise from the bottom up, with the left forest of a pair as the base. A pair meets at a seam, where the last page of the base meets the first page of the right forest. Every independent root of the right forest is placed into the base, and the rounds continue until one forest remains.

Only part of the base is offered as a target. We mark the roots of the base and the last node at every level of every subtree, and call these the active nodes. Every action of a round is taken against one of them. They are the nodes a following page can still continue, so the active set is the frontier of the base, and it spares a single call from carrying the whole forest. Each independent root of the right forest then receives one action, against one active node of the base. `merge` joins the root to the active node that is the same object, which is how a table continued across the seam becomes a single element with a box on each page it spans. `attach` places the root under the active node chosen as its parent, which is how a subsection beginning on the second page joins the section that begins

on the first. `keep` leaves the root at the top level, where a new section begins and the parent of the section is not in the base.

Merge is the action that carries identity, and it applies only where the right candidate is a single node. A region continued across the seam is the tail of an element, so it arrives with no children of its own, and this is where the regions of two pages become one element. It is also where a region becomes an element in the first place: a node turns into an element once something follows it, either a child beneath it or a later sibling, and a node with neither stays open along the right edge of the tree until a later round reaches it.

A system root closes the tree for serialization. It is not an element of the document and not a member of V: it carries no kind, no box and no text, and it is never a reference endpoint. Its children are the top-level elements of the document, every element of V^+ reaches it, and it is written once, after the last round, so the model never decides it.

In practice the model receives the active nodes of the base, the independent roots of the right forest, and the two boundary images at the seam. It emits one action per root, in the order of the roots. A merge names the active node it joins and an attach names the active node it goes under, each as a path of child indices in the base.

4.3 Reference linking

Reference linking populates E_r over the elements that forest construction has fixed. An anchor is an element carrying a reference. A target is an element a reference names, anywhere in the document. Pairing every element with every other is quadratic, and nearly all those pairs hold no reference. The step therefore settles the two roles first and matches across them. The anchor pool holds the elements carrying a reference marker: a cross-reference such as "see Figure 4", a citation, a superscript, or an implicit back-reference such as "as discussed in the previous chapter". The target pool holds the elements of $V^+ \cup V_{\text{ext}}$. Pairs are formed only inside a group, so the candidate set grows with the number and size of the groups rather than with the square of the element count.

Matching runs in bounded groups. Both pools are partitioned by the top-level subtree their elements belong to and by page, which keeps a reference to a distant appendix apart from a reference to the adjacent figure, and each group is cut to a fixed size, which caps the prompt of a single call. A pair is presented with the identifiers, categories, pages, coordinates and text summaries of its two candidates together with the relevant page images, and the model returns the pairs it accepts. Pairs from all groups are collected, deduplicated by their two identifiers, and validated: both endpoints exist, the anchor lies in a content element, and the two differ. The survivors become E_r, recorded as `link_to` on the anchor.

Writing a link leaves its target where it is. A target in the forest keeps its parent, and a target outside it stays beside the forest, so a reference can be a relation recorded next to the structure.

5 Experiments

Dataset **DocVine** is a manually annotated corpus of 1,217 full documents totalling 12,736 pages. Every document is annotated in two passes. The page pass marks each semantically complete element with its bounding box, category, and reading order. The document pass works across pages: it merges elements that continue over a page break, organizes the elements into ordered forest, and draws directed reference links between them. A group that carries no heading receives a virtual element, and the system root closes the tree. The annotations contain 43,581 non-leaf nodes and 124,680 hierarchy edges, together with 6,769 reference links. Cross-page identity is marked separately by 1,975 merged text instances and 825 merged tables. Of the non-leaf nodes, 12.97%, or 5,652, are virtual elements, so a substantial share of the hierarchy hangs on a node the document never prints. Documents are assigned to training, development, and test sets by a stable hash of the document identifier, giving 971, 155, and 91 documents. Composition and structural statistics are shown in Figure 3. More statistics can be found in Appendix B. The annotation guidelines are in Appendix C.

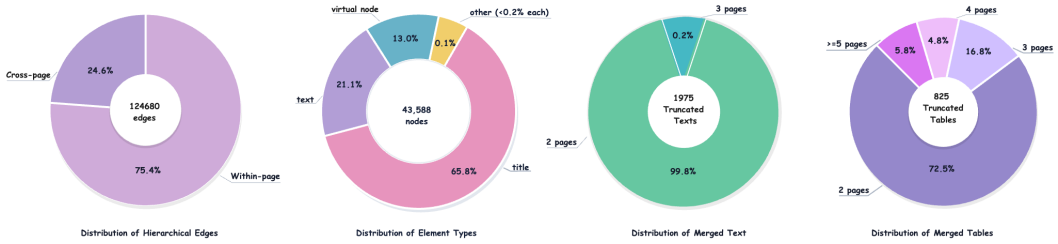


Figure 3: DocVine Statistics. From left to right, the proportions of within-page and cross-page hierarchy edges, together with the page-span distributions of 1,975 merged text instances and 825 merged tables. Percentages are calculated within each chart.

Table 1: Comparison on DocVine test set. *Direct* gives a model every page image of a document together with the OCR output of PaddleOCR-VL Cui et al. (2025), and output the whole structure at once. *VinePipe* uses the pipeline defined in Section 4. VinePipe + VineVLM-2B performs the best over all metrics.

Setting	Model	Soft-TEDS	Hierarchy Edge F1	Link F1	Merge F1
Direct	Qwen3.8-27B Qwen Team (2026)	42.4	29.7	44.0	54.2
	GLM-5.3-Flash Zhipu AI (2026)	28.0	22.1	42.9	35.3
	Kimi-K3 Moonshot AI (2026)	47.4	<u>37.9</u>	40.7	<u>66.0</u>
	DeepSeek-V4-Flash-Vision-Exp DeepSeek (2026)	39.4	33.2	41.8	47.5
	GPT-5.6-Sol OpenAI (2026)	27.2	21.5	41.8	18.1
	Claude-Fable-5 Anthropic (2026)	45.0	36.1	36.3	55.7
VinePipe	Qwen3.8-27B Qwen Team (2026)	49.1	35.9	<u>55.5</u>	51.6
	GLM-5.3-Flash Zhipu AI (2026)	27.2	20.5	37.8	20.0
	Kimi-K3 Moonshot AI (2026)	33.7	22.4	42.3	20.1
	DeepSeek-V4-Flash-Vision-Exp DeepSeek (2026)	41.5	26.5	46.8	41.9
	GPT-5.6-Sol OpenAI (2026)	24.7	18.3	34.1	2.4
	Claude-Fable-5 Anthropic (2026)	<u>52.0</u>	12.6	49.5	0.1
VinePipe	**VineVLM-2B**	**66.6**	**56.9**	**65.2**	**77.3**

Metrics Four numbers score the output $G = (V, E_h, E_r)$, and each reads it differently. *Soft-TEDS* measures the tree as a whole by tree edit distance between the predicted structure and the annotation, with soft node matches. *Hierarchy Edge F1* reads the parent of each element instead of the tree: an edge counts when an element hangs from the parent the annotation gives it, so this number measures grouping apart from how the regions were segmented. *Merge F1* measures whether fragments split by a page break are recovered as one element. *Link F1* counts a directed link only when its anchor and its target both agree with the annotation. The details of the metric definitions are in Appendix H.

Training We fine-tune Qwen3.5-2B with Megatron and call the output model **VineVLM-2B**. The vision encoder is frozen, while the language model and the vision language aligner are updated. We apply LoRA to all linear layers (rank 16, $\alpha = 32$). Training runs for 1 epoch at a learning rate of 1×10^{-4} with a global batch size of 32. The maximum sequence length is 81,920 tokens.

5.1 Comparison with generalist vision language models

We first compare our method with frontier vision language models on the DocVine test set. The models are Qwen3.8-27B Qwen Team (2026), GLM-5.3-Flash Zhipu AI (2026), Kimi-K3 Moonshot AI (2026), DeepSeek-V4-Flash-Vision-Exp DeepSeek (2026), GPT-5.6-Sol OpenAI (2026), and Claude-Fable-5 Anthropic (2026), and each is run under two settings. *Direct* gives the model all page images of a document together with the OCR output of PaddleOCR-VL Cui et al. (2025), and asks for the whole structure at once. *VinePipe* runs the same model through the stages defined in Section 4. Every method here is built on PaddleOCR-VL, which supplies the per-page OCR. As shown in Table 1, VineVLM-2B ranks first on all four metrics, at 66.6 Soft-TEDS, 56.9 Hierarchy Edge F1, 65.2 Link F1, and 77.3 Merge F1. The next-best Soft-TEDS is VinePipe+Claude-Fable-5 at 52.0. VinePipe+Qwen3.8-27B remains the strongest prompted row on Link F1, and every other row is behind on all four metrics by at least 9.7.

Table 2: Capability-aligned comparison with representative document parsing methods. Each block is read only against its own baseline, since the projections differ. Configurations are in Appendix I.

Method	Soft TEDS	Hierarchy Edge F1	Link F1	Merge F1
DOSA Li et al. (2026)	59.1	66.7	–	–
VineVLM-2B	**80.7**	**74.9**	**99.5**	–
DocParser Rausch et al. (2021)	9.6	6.7	–	–
Docling Livathinos et al. (2025)	19.6	4.5	–	–
TextIn TextIn (2025)	36.7	13.1	–	46.5
MinerU-Popo Xu et al. (2026a)	45.6	30.4	44.0	54.9
VineVLM-2B	**68.1**	**59.2**	**65.2**	**77.3**

The advantage comes from the pipeline and the supervision together, and the pipeline's contribution differs by setting. Running through it without training, the frontier base models trade one metric for another. Training is where the pipeline earns its place. Working stage by stage means the model never has to hold a whole document at once. A direct whole-document objective would place the entire serialized structure in one context, and under a budget of 81,920 tokens the longest documents would drop out of the corpus and fewer sequences would fit in a batch, so that objective is both harder to train and narrower in coverage. Stage-wise supervision reaches those documents instead, because a long document whose structure exceeds the budget is still supervised stage by stage.

5.2 Comparison with task-specific parsers

Next we compare with task-specific parsers. These are DOSA Li et al. (2026), DocParser Rausch et al. (2021), Docling Livathinos et al. (2025), TextIn TextIn (2025), and MinerU-PoPo Xu et al. (2026a). None of them predicts the full structure of G, so each is compared under the representation it can express: both the DocVine annotation and the VineVLM-2B prediction are projected into that system's label and structural space, on the DocVine test set. The blocks of Table 2 state their own protocol in their header row and are read only against the baseline of their own block.

DOSA trains a post-processing model over ground-truth elements. We run it as released and train it on DocVine, and to keep the comparison fair VinePipe with VineVLM-2B receives the same ground-truth elements, which is the first block of the table. Both sides then work with oracle elements, and virtual elements and cross-page merges are removed to match the scope DOSA expresses. Thus the scores in this block are relatively higher than those in Table 1.

DocParser and Docling are trained on other corpora and only their weights are released. DocParser comes from arXivdocs, a corpus of English scientific papers rebuilt by reverse rendering LaTeX source, and neither has seen a corpus that spans document types as DocVine does. Applied without any DocVine fine-tuning, both score extremely low, so these rows measure transfer rather than either method: a parser fitted to one rendering pipeline and one genre does not carry beyond them. The projection is applied identically to both sides and contributes nothing to the gap.

TextIn is reached through an interface alone and MinerU-PoPo is an open vision language model, and both do better than the two systems above, at 36.7 and 45.6 Soft TEDS against 9.6 and 19.6. The reason is scope: these two are the more general systems of the group, and they are the ones that also combine fragments across pages. None of the five methods goes past a heading hierarchy, so for Soft TEDS and Hierarchy Edge F1 we keep only the heading part of DocVine and remove virtual elements from both sides, which is why our numbers here are slightly higher than in Table 1. Keeping the virtual elements would lower every one of these baselines, since a heading hierarchy has nowhere to put the groups those elements host. On every metric reported in this table, VineVLM-2B is the best.

5.3 Evaluation on MPDocBench-Parse

To further validate our corpus and method, we run VinePipe with VineVLM-2B on the recently proposed MPDocBench-Parse Zhou et al. (2026) benchmark. We report the results in Table 3. All the scores except ours are taken from the original paper. Our method achieves the highest overall

Table 3: Overall Performance of Document Parsing Methods on MPDocBench-Parse. T-Txt ED: Truncated-Text Edit Distance; T-Tbl TEDS: Truncated-Table TEDS; R-O ED: Reading-Order Edit Distance; T-TEDS: Title TEDS.

Category	Model	Overall Score	T-Txt ED↓	T-Tbl TEDS↑	R-O ED↓	Title TEDS↑
Pipeline-Based Specialized VLM	GLM-OCR Duan et al. (2026)	66.00	0.314	63.10	0.125	44.79
	PaddleOCR-VL-1.5 Cui et al. (2026)	<u>76.15</u>	<u>0.151</u>	83.41	0.108	47.10
	MinerU2.5 Niu et al. (2025)	70.06	0.328	**88.14**	0.120	36.90
	Youtu-Parsing Yin et al. (2026)	64.75	0.353	63.88	0.132	43.64
	MonkeyOCR-pro-3B Zhang et al. (2025)	64.88	0.300	61.17	0.120	40.34
	Dolphin-v2 Feng et al. (2026)	66.81	0.334	64.28	0.137	50.04
End-to-End Specialized VLM	dots.mocr Zheng et al. (2026)	63.73	0.298	62.16	0.111	33.68
	FireRed-OCR Wu et al. (2026)	71.77	0.178	62.62	<u>0.086</u>	<u>50.84</u>
	dots.ocr Li et al. (2025)	66.02	0.303	60.68	0.114	45.12
	DeepSeek-OCR2 Wei et al. (2026)	68.38	0.284	63.01	0.106	49.53
	OCRVerse Zhong et al. (2026)	63.83	0.291	63.96	0.150	35.45
	Logics-Parsing-v2 Chen et al. (2025b)	64.69	0.312	63.86	**0.085**	34.62
	Qianfan-OCR Dong et al. (2026)	63.22	0.474	62.41	0.113	49.19
	ChandraOCR 2 Datalab (2025)	67.80	0.288	64.67	0.133	48.64
General-Purpose VLM	Gemini-3.1-pro-preview Google DeepMind (2026)	63.08	0.232	61.26	0.127	26.94
	ChatGPT-5.2-2025-12-11 OpenAI (2025)	60.53	0.372	58.71	0.169	37.50
	Qwen3.6-plus TongYi (2025)	62.54	0.245	60.50	0.179	32.06
	Qwen3-VL-235B Bai et al. (2025)	67.82	0.187	61.25	0.136	42.34
	InternVL-3.5-38B Wang et al. (2025)	51.82	0.515	51.69	0.199	27.01
	VineVLM-2B	**78.83**	**0.100**	<u>84.29</u>	0.143	**55.30**

score. Its Title-TEDS stands above every other system, at 55.30 against 50.84 for the next. This is the clearest evidence we have that the method understands section structure, and it comes from a metric that scores only a degenerate version of that structure, since a heading hierarchy carries no virtual elements and no reference links. The overall lead also rests on the two truncated metrics, where we rank first on truncated text edit distance and second on truncated table TEDS, which shows that element merging holds up across page boundaries.

Reading order is the one component where we do not lead, and the gap is a consequence of the definition rather than of the method. Documents contain many float elements. A system that orders blocks by where they appear on the page places such an element in the reading sequence at its page position, but a float is not read there. Our method keeps these elements in V_{ext} and attaches them to the main text through reference links, so their page position never enters the reading order it predicts. The two protocols therefore disagree about what the sequence should contain before they disagree about the order of it. We also record the per-category Title-TEDS scores in Appendix E, demonstrating the same standing across document types rather than in one category alone.

6 Conclusion

In this paper we widened document structure understanding from a heading hierarchy to the prediction of one structure $G = (V, E_h, E_r)$ over a document. A heading hierarchy is a degenerate case of this problem. We instantiate the definition with DocVine, a corpus of 1,217 documents and 12,736 pages, and with VinePipe, a pipeline driven end to end by a fine-tuned 2B vision language model, VineVLM-2B. On DocVine the model surpasses task-specific hierarchy parsers and prompted frontier vision language models, and it leads on MPDocBench-Parse under that benchmark's own protocol, indicating the wider formulation benefits the narrower task. Beyond the aggregate scores, the case studies in Appendix A show why our definition and method perform better on the DocVine test set.

AI USE STATEMENT

We used generative AI tools to assist with the writing and the presentation of this paper, including drafting and editing prose, restructuring sections. We used them as well to assist in implementing parts of the VinePipe codebase, the model training, and the metric implementations.

All AI-assisted work was reviewed by the authors. Prose edited with these tools was checked against the underlying results, and every number reported here was computed from the released code and the frozen test document list. Code written with their assistance was reviewed by the authors, exercised by unit tests, and run end to end on a held-out set of documents before any result was recorded.

DocVine was annotated by human annotators under the guidelines in Appendix C.

We take responsibility for the final content of this work, including text, claims and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

DocVine is assembled from publicly available documents of the categories listed in Appendix B, and we will release it for the community. The corpus was annotated by paid annotators who were informed of the purpose of the work and of how the annotations would be released. Annotators were recruited contracted, compensated at a reasonable salary, and are not identified in this paper.

REPRODUCIBILITY STATEMENT

We decide to release our dataset, training protocal, the fintuned model and the inference pipeline after the reviewing process.

Training. §5 gives the base model, the frozen vision encoder, the LoRA configuration (rank 16, $\alpha = 32$), the learning rate, the global batch size, the epoch count and the maximum sequence length.

Evaluation. Every score follows one protocol: the fixed DocVine test set, macro-averaging with missing or failed predictions scored zero, and the metric definitions in Appendix H, with the link alignment scope in Appendix H.4. The capability-aligned projections of §5.2 are specified in Appendix I, and the MPDocBench category results in Appendix E. We report bootstrap intervals over the test documents for the margins in §5.1.

Baselines. DocParser, Docling, TextIn, MinerU-PoPo and the generalist models are used as released or through their public interfaces. DOSA has no public code, so we reimplemented it from its paper and trained it on DocVine. The reimplementation is part of the release, and its deviations from the published description are listed in Appendix I.

REFERENCES

Anthropic. Claude Fable 5 and Claude Mythos 5. `https://www.anthropic.com/news/claude-fable-5-mythos-5`, 2026.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Peng Cai, Zhaofan Zou, Shifa Liu, Yikun Wang, Jiawei Tang, Kaicheng Yang, Meng Tong, MingKun Jiang, Zhongjiang He, and Hao Sun. NaviDC-OCR: Navigating document parsing across digital and camera-captured documents. *arXiv preprint arXiv:2608.12898*, 2026.

Jian Chen, Ruiyi Zhang, Yufan Zhou, Tong Yu, Franck Dernoncourt, Jiuxiang Gu, Ryan A. Rossi, Changyou Chen, and Tong Sun. SV-RAG: LoRA-contextualizing adaptation of MLLMs for long document understanding. In *International Conference on Learning Representations (ICLR)*, 2025a.

Xiangyang Chen, Shuzhao Li, Xiuwen Zhu, Yongfan Chen, Fan Yang, Cheng Fang, Lin Qu, Xiaoxiao Xu, Hu Wei, and Minggang Wu. Logics-parsing technical report. *arXiv preprint arXiv:2509.19760*, 2025b.

Yufan Chen, Ruiping Liu, Junwei Zheng, Di Wen, Kunyu Peng, Jiaming Zhang, and Rainer Stiefelhagen. Graph-based document structure analysis. In *International Conference on Learning Representations (ICLR)*, 2025c.

Jaemin Cho, Debanjan Mahata, Ozan Irsoy, Yujie He, and Mohit Bansal. M3DocRAG: Multimodal retrieval is what you need for multi-page multi-document understanding. *arXiv preprint arXiv:2411.04952*, 2024.

Cheng Cui, Ting Sun, Suyin Liang, Tingquan Gao, Zelun Zhang, Jiaxuan Liu, Xueqing Wang, Changda Zhou, Hongen Liu, Manhui Lin, Yue Zhang, Yubo Zhang, Handong Zheng, Jing Zhang, Jun Zhang, Yi Liu, Dianhai Yu, and Yanjun Ma. PaddleOCR-VL: Boosting multilingual document parsing via a 0.9b ultra-compact vision-language model. *arXiv preprint arXiv:2510.14528*, 2025.

Cheng Cui, Ting Sun, Suyin Liang, Tingquan Gao, Zelun Zhang, Jiaxuan Liu, Xueqing Wang, Changda Zhou, Hongen Liu, Manhui Lin, Yue Zhang, Yubo Zhang, Yi Liu, Dianhai Yu, and Yanjun Ma. PaddleOCR-VL-1.5: Towards a multi-task 0.9B VLM for robust in-the-wild document parsing. *arXiv preprint arXiv:2601.21957*, 2026.

Lei Cui, Yiheng Xu, Tengchao Lv, and Furu Wei. Document AI: Benchmarks, models and applications. *arXiv preprint arXiv:2111.08609*, 2021.

Datalab. Chandra OCR 2. `https://github.com/datalab-to/chandra`, 2025.

DeepSeek. DeepSeek-V4-Flash-Vision-Exp release. `https://api-docs.deepseek.com/news/news260821/`, 2026.

Daxiang Dong, Mingming Zheng, Dong Xu, Chunhua Luo, Bairong Zhuang, Yuxuan Li, Ruoyun He, Haoran Wang, Wenyu Zhang, Wenbo Wang, et al. Qianfan-OCR: A unified end-to-end model for document intelligence. *arXiv preprint arXiv:2603.13398*, 2026.

Shuaiqi Duan, Yadong Xue, Weihan Wang, Zhe Su, Huan Liu, Sheng Yang, Guobing Gan, Guo Wang, et al. GLM-OCR technical report. *arXiv preprint arXiv:2603.10910*, 2026.

Hao Feng, Wei Shi, Ke Zhang, Xiang Fei, Lei Liao, Dingkang Yang, Yongkun Du, Xuecheng Wu, Jingqun Tang, Yang Liu, Hong Chen, and Can Huang. Dolphin-v2: Universal document parsing via scalable anchor prompting. *arXiv preprint arXiv:2602.05384*, 2026.

Google DeepMind. Gemini 3.1. `https://deepmind.google/models/gemini/pro/`, 2026.

Yupan Huang, Tengchao Lv, Lei Cui, Yutong Lu, and Furu Wei. LayoutLMv3: Pre-training for document AI with unified text and image masking. In *Proceedings of the 30th ACM International Conference on Multimedia*, pp. 4083–4091, 2022.

Guillaume Jaume, Hazim Kemal Ekenel, and Jean-Philippe Thiran. FUNSD: A dataset for form understanding in noisy scanned documents. In *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*, pp. 1–6, 2019. doi: 10.1109/ICDARW.2019.10029.

Bohou Li, Benjamin Sowell, Mehul Shah, Mark Lindblad, and Henry Lindeman. DOSA: A tree-guided, self-regressive framework for long document structure analysis. *arXiv preprint arXiv:2607.22679*, 2026.

Minghao Li, Yiheng Xu, Lei Cui, Shaohan Huang, Furu Wei, Zhoujun Li, and Ming Zhou. DocBank: A benchmark dataset for document layout analysis. In *Proceedings of the 28th International Conference on Computational Linguistics (COLING)*, pp. 949–960, 2020. doi: 10.18653/v1/2020.coling-main.82.

Yumeng Li, Guang Yang, Hao Liu, Bowen Wang, and Colin Zhang. dots.ocr: Multilingual document layout parsing in a single vision-language model. *arXiv preprint arXiv:2512.02498*, 2025.

Yiming Lin, Madelon Hulsebos, Ruiying Ma, Shreya Shankar, Sepanta Zeighami, Aditya G. Parameswaran, and Eugene Wu. Towards accurate and efficient document analytics with large language models. *arXiv preprint arXiv:2405.04674*, 2024.

Nikos Livathinos, Christoph Auer, Maxim Lysak, Ahmed Nassar, Michele Dolfi, Panos Vagenas, Cesar Berrospi Ramis, Matteo Omenetti, Kasper Dinkla, Yusik Kim, et al. Docling: An efficient open-source toolkit for AI-driven document conversion. In *AAAI Conference on Artificial Intelligence*, 2025.

António Loison, Quentin Macé, Antoine Edy, Victor Xing, Tom Balough, Gabriel Moreira, Bo Liu, Manuel Faysse, Céline Hudelot, and Gautier Viaud. ViDoRe v3: A comprehensive evaluation of retrieval augmented generation in complex real-world scenarios. *arXiv preprint arXiv:2601.08620*, 2026.

Patrice Lopez. GROBID: Combining automatic bibliographic data recognition and term extraction for scholarship publications. In *Proceedings of the 13th European Conference on Digital Libraries (ECDL)*, pp. 473–474, 2009.

Shiyin Lu, Yinglun Li, Yu Xia, Yuhui Chen, An-Yang Ji, Jun-Peng Jiang, Qing-Guo Chen, Jianshan Zhao, En Lin, Haijun Li, Cheng Qin, Zhao Xu, and Weihua Luo. OvisOCR2 technical report. *arXiv preprint arXiv:2607.13639*, 2026.

Jiefeng Ma, Jun Du, Pengfei Hu, Zhenrong Zhang, Jianshu Zhang, Huihui Zhu, and Cong Liu. HRDoc: Dataset and baseline method toward hierarchical reconstruction of document structures. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 1870–1877, 2023.

Moonshot AI. Kimi K3: Open frontier intelligence. `https://www.kimi.ai/blog/kimi-k3`, 2026.

George Nagy and Sharad Seth. Hierarchical representation of optically scanned documents. In *Proceedings of the 7th International Conference on Pattern Recognition (ICPR)*, pp. 347–349, 1984.

Junbo Niu, Zheng Liu, Zhuangcheng Gu, Bin Wang, Linke Ouyang, Zhiyuan Zhao, Tao Chu, Tianyao He, et al. MinerU2.5: A decoupled vision-language model for efficient high-resolution document parsing. *arXiv preprint arXiv:2509.22186*, 2025.

Lawrence O'Gorman. The document spectrum for page layout analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1162–1173, 1993.

OpenAI. ChatGPT. `https://chat.openai.com`, 2025.

OpenAI. GPT-5.6: Frontier intelligence that scales with your ambition. `https://openai.com/index/gpt-5-6/`, 2026.

Linke Ouyang, Yuan Qu, Hongbin Zhou, Jiawei Zhu, Rui Zhang, Qunshu Lin, Bin Wang, Zhiyuan Zhao, Man Jiang, Xiaomeng Zhao, et al. OmniDocBench: Benchmarking diverse PDF document parsing with comprehensive annotations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Birgit Pfitzmann, Christoph Auer, Michele Dolfi, Ahmed S. Nassar, and Peter W. J. Staar. DocLayNet: A large human-annotated dataset for document-layout analysis. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 374–382, 2022.

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

Qwen Team. Qwen3.8. `https://qwen.ai/blog?id=qwen3.8`, 2026.

Johannes Rausch, Octavio Martinez, Fabian Bissig, Ce Zhang, and Stefan Feuerriegel. Docparser: Hierarchical document structure parsing from renderings. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

Johannes Rausch, Gentiana Rashiti, Maxim Gusev, Ce Zhang, and Stefan Feuerriegel. Dsg: An end-to-end document structure generator. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, 2023.

Jon Saad-Falcon, Joe Barrow, Alexa Siu, Ani Nenkova, David Seunghyun Yoon, Ryan A. Rossi, and Franck Dernoncourt. PDFTriage: Question answering over long, structured documents. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 153–169, 2024.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. RAPTOR: Recursive abstractive processing for tree-organized retrieval. In *International Conference on Learning Representations (ICLR)*, 2024.

Zejiang Shen, Ruochen Zhang, Melissa Dell, Benjamin Charles Germain Lee, Jacob Carlson, and Weining Li. LayoutParser: A unified toolkit for deep learning based document image analysis. In *Document Analysis and Recognition – ICDAR 2021*, pp. 131–146, 2021. doi: 10.1007/978-3-030-86549-8_9.

Joongmin Shin, Chanjun Park, Jeongbae Park, Jaehyung Seo, and Heuiseok Lim. MultiDocFusion: Hierarchical and multimodal chunking pipeline for enhanced RAG on long industrial documents. *arXiv preprint arXiv:2604.12352*, 2026a.

Joongmin Shin, Jeongbae Park, Jaehyung Seo, and Heuiseok Lim. M3DocDep: Multi-modal, multi-page, multi-document dependency chunking with large vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026b.

Joongmin Shin, Gyuho Shim, Jeongbae Park, Jaehyung Seo, and Heuiseok Lim. HiKEY: Hierarchical multimodal retrieval for open-domain document question answering. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 17967–17987, 2026c.

Ray Smith. An overview of the tesseract OCR engine. In *Proceedings of the Ninth International Conference on Document Analysis and Recognition (ICDAR)*, volume 2, pp. 629–633, 2007.

Brandon Smock, Rohith Pesala, and Robin Abraham. PubTables-1M: Towards comprehensive table extraction from unstructured documents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4634–4642, 2022.

Zineng Tang, Ziyi Yang, Guoxin Wang, Yuwei Fang, Yang Liu, Chenguang Zhu, Michael Zeng, Cha Zhang, and Mohit Bansal. Unifying vision, text, and layout for universal document processing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Tencent Hunyuan Vision Team. HunyuanOCR technical report. *arXiv preprint arXiv:2511.19575*, 2025.

TextIn. Textin. `https://www.textin.com/`, 2025.

TongYi. Tongyi. `https://www.aliyun.com/product/tongyi`, 2025.

Bin Wang, Tianyao He, Linke Ouyang, Fan Wu, Zhiyuan Zhao, Tao Chu, Zhenjiang Jin, Weijun Zeng, et al. MinerU2.5-Pro: Pushing the limits of data-centric document parsing at scale. *arXiv preprint arXiv:2604.04771*, 2026.

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, et al. InternVL3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *arXiv preprint arXiv:2508.18265*, 2025.

Zilong Wang, Mingjie Zhan, Xuebo Liu, and Ding Liang. DocStruct: A multimodal method to extract hierarchy structure in document for general form understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 898–908, 2020. doi: 10.18653/v1/2020.findings-emnlp.80.

Zilong Wang, Yiheng Xu, Lei Cui, Jingbo Shang, and Furu Wei. LayoutReader: Pre-training of text and layout for reading order detection. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 4735–4744, 2021.

Haoran Wei, Yaofeng Sun, and Yukun Li. DeepSeek-OCR: Contexts optical compression. *arXiv preprint arXiv:2510.18234*, 2025.

Haoran Wei, Yaofeng Sun, and Yukun Li. DeepSeek-OCR 2: Visual causal flow. *arXiv preprint arXiv:2601.20552*, 2026.

Hao Wu, Haoran Lou, Xinyue Li, Zuodong Zhong, Zhaojun Sun, Phellon Chen, Xuanhe Zhou, Kai Zuo, Yibo Chen, Xu Tang, et al. FireRed-OCR technical report. *arXiv preprint arXiv:2603.01840*, 2026.

Renqiu Xia, Song Mao, Xiangchao Yan, Hongbin Zhou, Bo Zhang, Haoyang Peng, Jiahao Pi, Daocheng Fu, Wenjie Wu, Hancheng Ye, Shiyang Feng, Bin Wang, Chao Xu, Conghui He, Pinlong Cai, Min Dou, Botian Shi, Sheng Zhou, Yongwei Wang, Bin Wang, Junchi Yan, Fei Wu, and Yu Qiao. Docgenome: An open large-scale scientific document benchmark for training and testing multi-modal large language models. *arXiv preprint arXiv:2406.11633*, 2024.

Hangdi Xing, Changxu Cheng, Feiyu Gao, Zirui Shao, Zhi Yu, Jiajun Bu, Qi Zheng, and Cong Yao. DocHieNet: A large and diverse dataset for document hierarchy parsing. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1129–1142, 2024.

Bangrui Xu, Ziyang Miao, Xuanhe Zhou, Yiming Lin, Zirui Tang, Xiaomeng Zhao, Fan Wu, Cheng Tan, Bin Wang, and Conghui He. MinerU-PoPo: Universal post-processing model for structured document parsing. *arXiv preprint arXiv:2605.24973*, 2026a.

Bangrui Xu, Qihang Yao, Zirui Tang, Xuanhe Zhou, Yeye He, Shihan Yu, Qianqian Xu, Bin Wang, Guoliang Li, Conghui He, and Fan Wu. MoDora: Tree-based semi-structured document analysis system. *arXiv preprint arXiv:2602.23061*, 2026b.

Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming 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*, pp. 1192–1200, 2020.

Kun Yin, Yunfei Wu, Bing Liu, Zhongpeng Cai, Xiaotian Li, Huang Chen, Xin Li, Haoyu Cao, Yinsong Liu, Deqiang Jiang, et al. Youtu-Parsing: Perception, structuring and recognition via high-parallelism decoding. *arXiv preprint arXiv:2601.20430*, 2026.

Jiarui Zhang, Yuliang Liu, Zijun Wu, Guosheng Pang, Zhili Ye, Yupei Zhong, Junteng Ma, Tao Wei, Haiyang Xu, Weikai Chen, Zeen Wang, Qiangjun Ji, Fanxi Zhou, Qi Zhang, Yuanrui Hu, Jiahao Liu, Zhang Li, Ziyang Zhang, Qiang Liu, and Xiang Bai. MonkeyOCR v1.5 technical report: Unlocking robust document parsing for complex patterns. *arXiv preprint arXiv:2511.10390*, 2025.

Yue Zhang, Zhihao Zhang, Wenbin Lai, Chong Zhang, Tao Gui, Qi Zhang, and Xuanjing Huang. PDF-to-Tree: Parsing PDF text blocks into a tree. In *Findings of the Association for Computational Linguistics: EMNLP*, pp. 10704–10714, 2024.

Handong Zheng, Yumeng Li, Kaile Zhang, Liang Xin, Guangwei Zhao, Hao Liu, Jiayu Chen, Jie Lou, Jiyu Qiu, Qi Fu, et al. Multimodal OCR: Parse anything from documents. *arXiv preprint arXiv:2603.13032*, 2026.

Zhipu AI. GLM-5.3-Flash: Frontier intelligence, flash cost. `https://z.ai/blog/glm-5.3-flash`, 2026.

Xu Zhong, Jianbin Tang, and Antonio Jimeno Yepes. PubLayNet: Largest dataset ever for document layout analysis. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pp. 1015–1022, 2019. doi: 10.1109/ICDAR.2019.00166.

Yufeng Zhong, Lei Chen, Xuanle Zhao, Wenkang Han, Liming Zheng, Jing Huang, Deyang Jiang, Yilin Cao, Lin Ma, and Zhixiong Zeng. OCRVerse: Towards holistic OCR in end-to-end vision-language models. *arXiv preprint arXiv:2601.21639*, 2026.

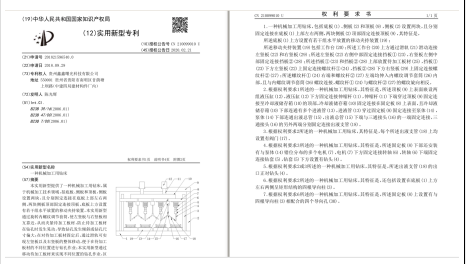
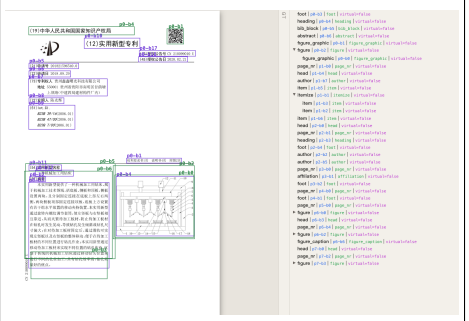
Bangbang Zhou, Hangdi Xing, Yifan Chen, Jianjun Xu, Qi Zheng, Feiyu Gao, Zhibo Yang, Shuai Bai, Ming Yan, Jieping Ye, and Hongtao Xie. Mpdocbench-parse: Benchmarking practical multi-page document parsing. *arXiv preprint arXiv:2605.22100*, 2026.

Yuefeng Zou, Yichen Lu, Jingxiao Yang, Bingtao Fu, Gaoyang Zhang, Xiongfei Bai, Tian Chen, and Xiang Qi. LongDocBench: Benchmarking TOC hierarchy and contextual relationship recovery in long documents. *arXiv preprint arXiv:2608.15064*, 2026.

A CASE STUDY

A.1 COMPARISON WITH RELATED METHODS

Table 4: Qualitative comparison on the first pair of examples. **Left example:**DOSA produces finer-grained false positives; DocParser misses detection boxes; Docling-Granite misses many headings; TextIn misses headings; and MinerU-Popo misses headings and produces an incorrect image reading order. **Right example:** DOSA predicts an incorrect heading hierarchy; DocParser misses detection boxes; Docling-Granite predicts an incorrect heading hierarchy; TextIn repeatedly includes non-body content; and MinerU-Popo does not complete some cross-page merges.

Original		
DOSA		
DocParser		

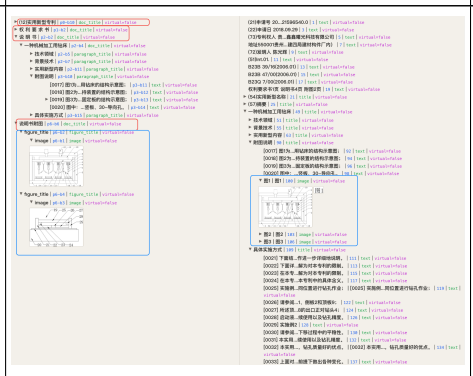
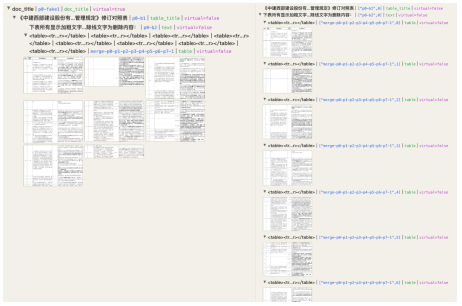
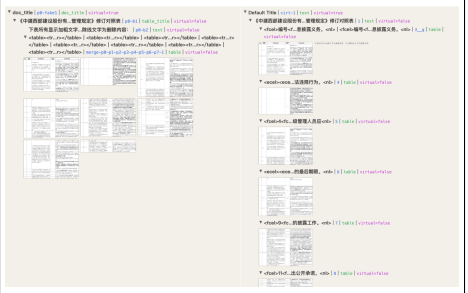
Docling-Granite		
TextIn		
MinerU-Popo		
VineVLM		

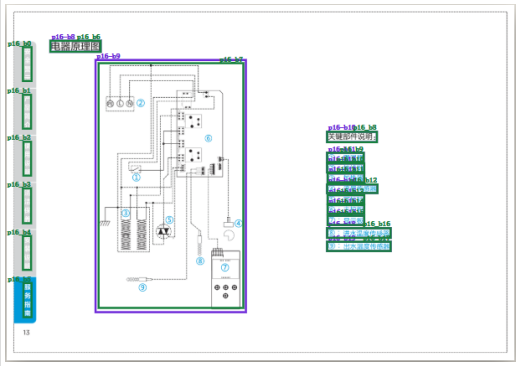
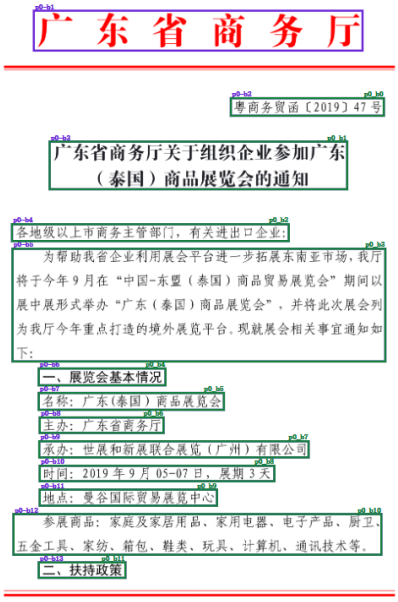
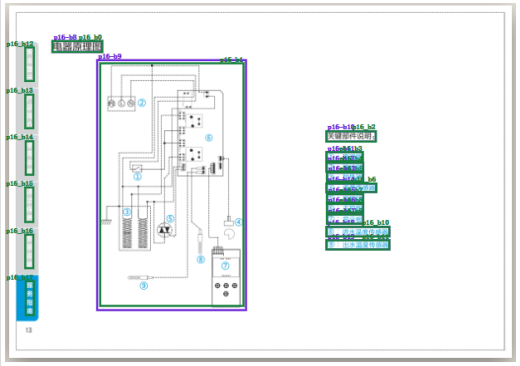
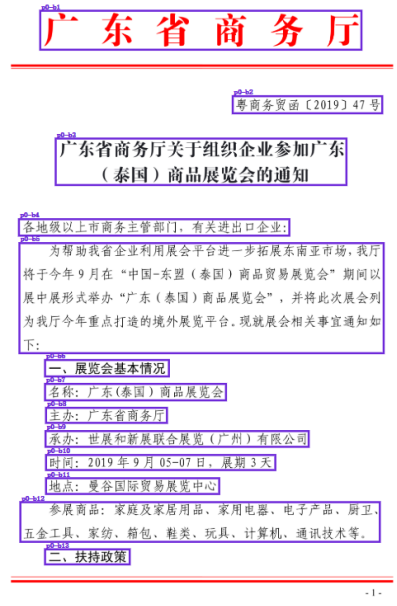
Table 5: Qualitative comparison on the second pair of examples. **Left example:** DOSA does not merge the table across pages; DocParser produces disordered bounding-box detections; Docling-Granite recognizes the table incorrectly; TextIn performs well; and MinerU-Popo does not merge the table across pages. **Right example:** DocParser produces disordered bounding-box detections; Docling-Granite detects the content as a single bounding box; MinerU-Popo misses detection boxes and does not recover the hierarchy; and VineVLM misses small-font headings and produces an ambiguous interpretation of the image.

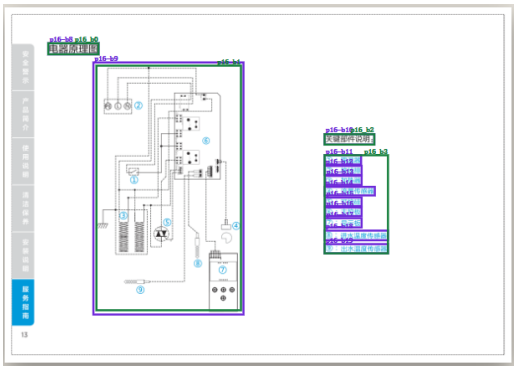
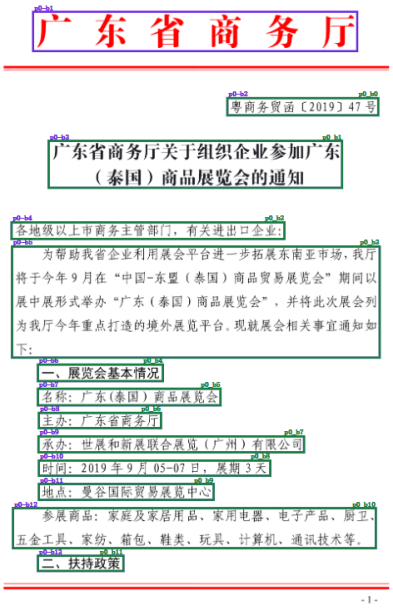
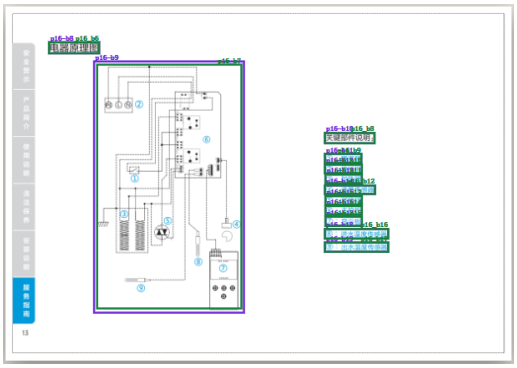
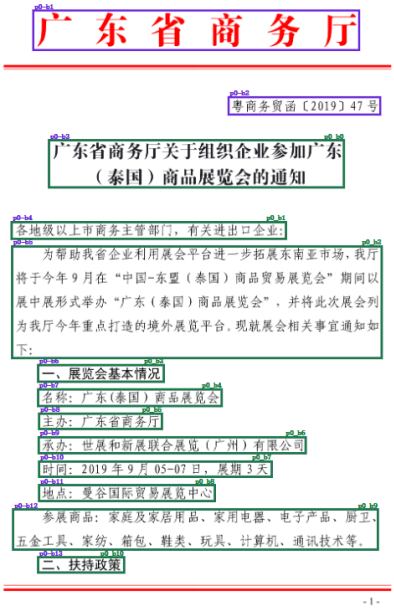
Original		
DOSA		
DocParser		

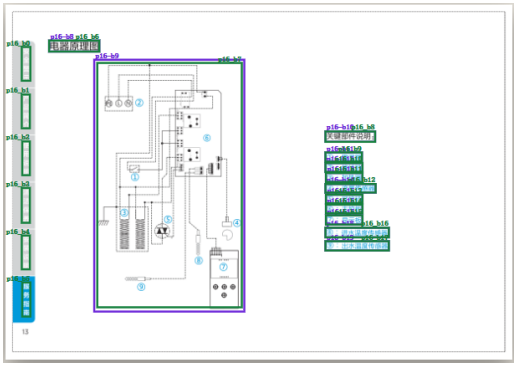
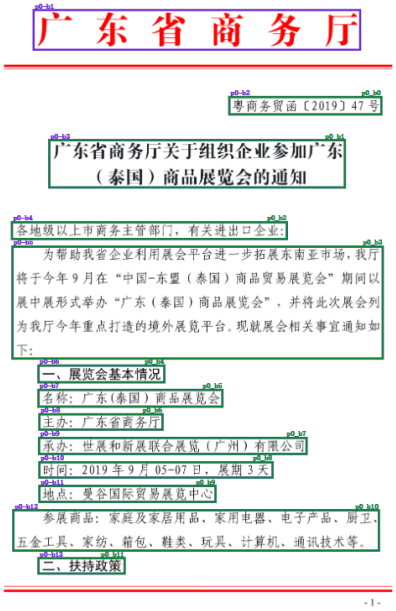
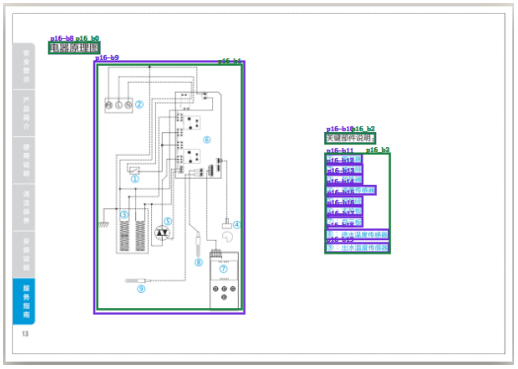
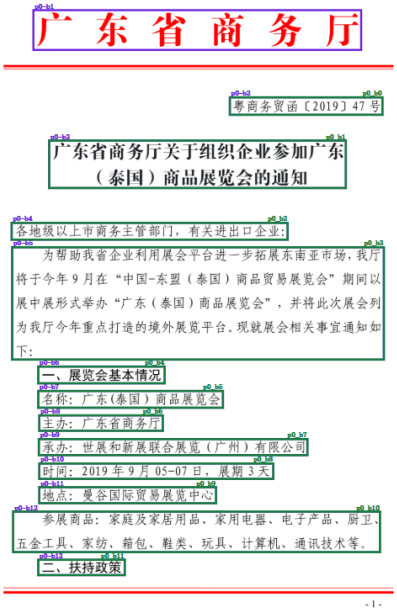
Docling-Granite		
TextIn		
MinerU-Popo		
VineVLM		

A.2 Qualitative Examples of Joint DSU Predictions

Table 6: Qualitative comparison on two semantic-block cases. **Left, false-positive removal:** Ours removes the spurious label box on the left while preserving all valid content regions. Most competing methods retain the false detection. **Right, missed OCR region:** Ours restores the omitted "Guangdong Department of Commerce" heading and aligns it with the ground truth, while baselines tend to omit or mislocalize it. —— Ground truth —— Prediction.

Qwen3.8		
GLM-5.3		

Kimi-K3		
DeepSeek-V4.1		

GPT-5.6-sol		
Claude-Fable-5		

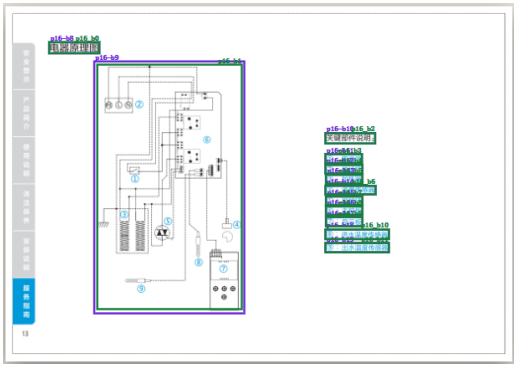
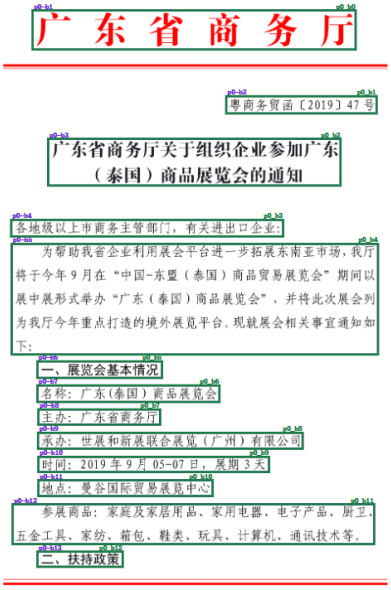
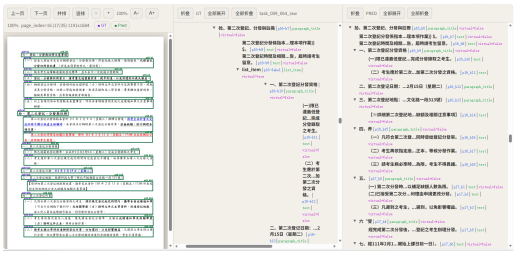
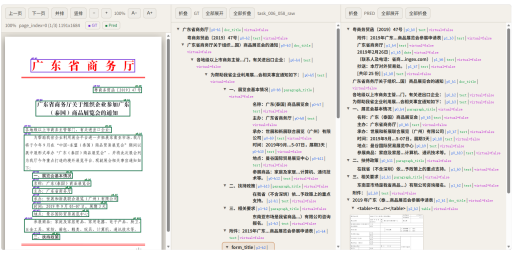
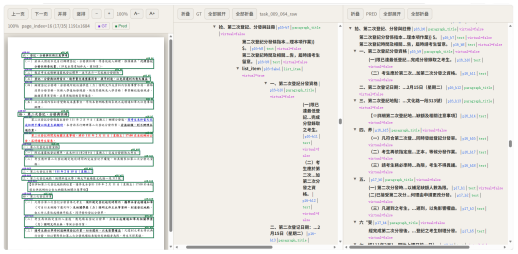
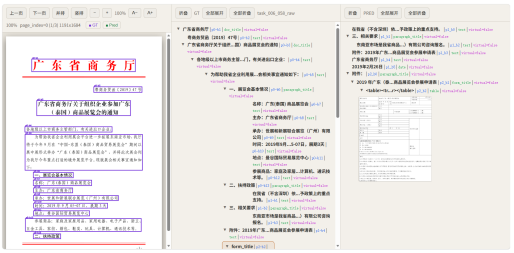
Ours		

Table 7: Qualitative comparison on two structure cases. **Left, virtual `list_title`:** Ours creates a concise virtual `list_title` that summarizes the following items and exposes their shared parent. Baselines more often flatten the list or select an incorrect grouping. **Right, one item with multiple paragraphs:** Ours groups all continuation paragraphs under the same item even without an explicit title style. Competing methods tend to split the multi-paragraph unit or flatten its parent–child relation. Ground truth Prediction.

Qwen3.8		
GLM-5.3		

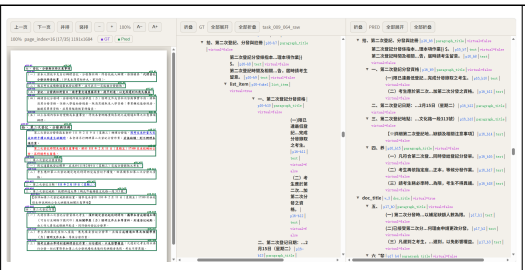
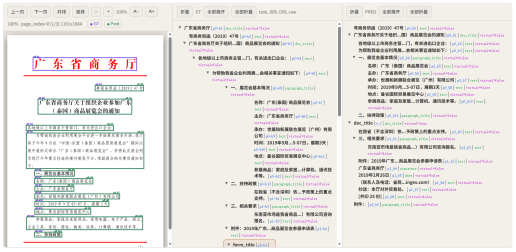
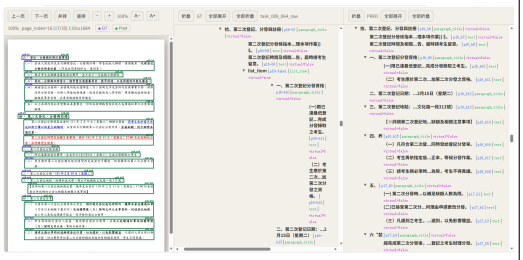
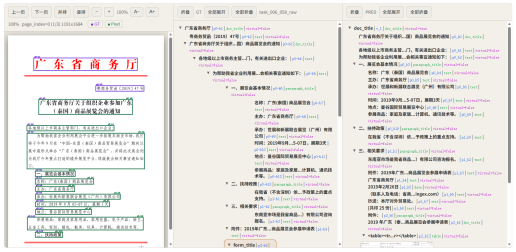
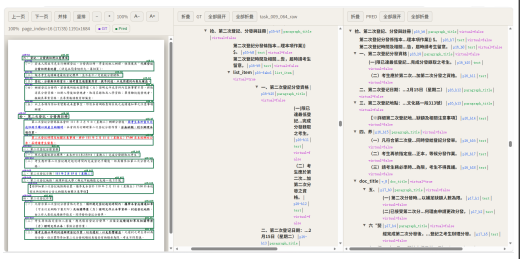
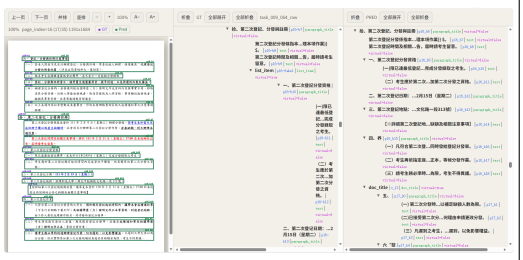
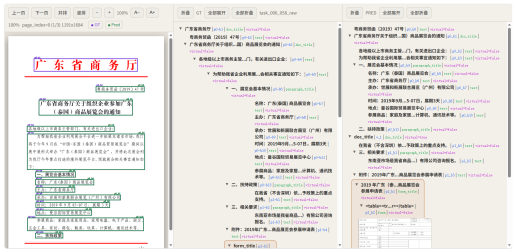
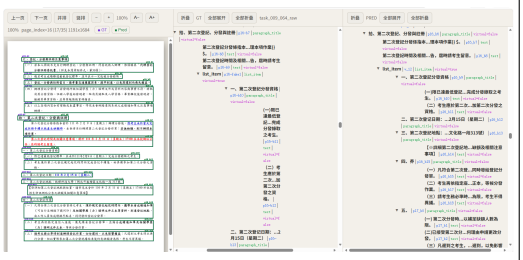
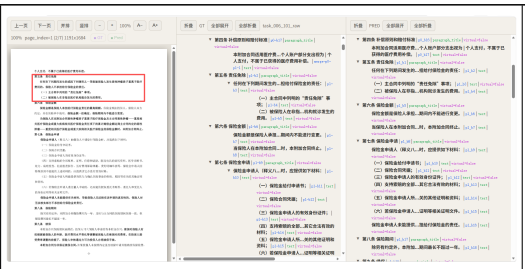
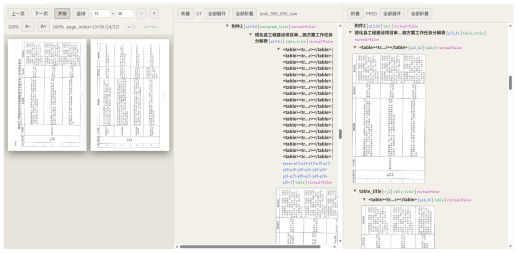
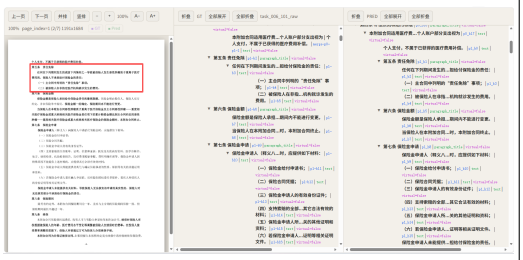
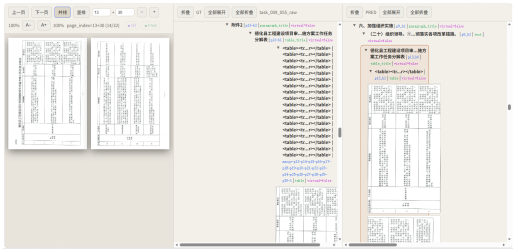
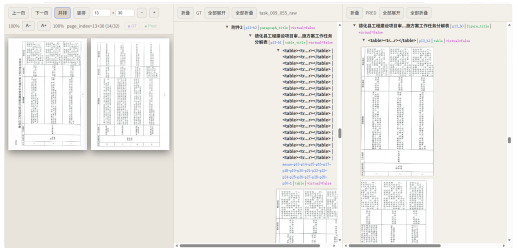
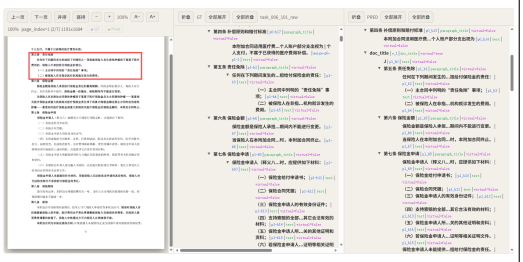
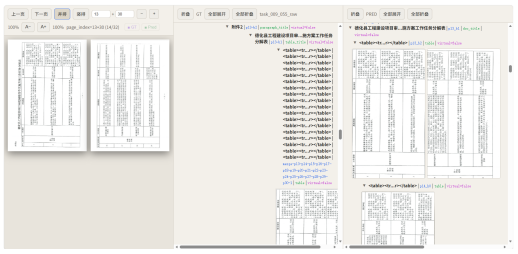
Kimi-K3		
DeepSeek-V4.1		
GPT-5.6-sol		
Claude-Fable-5		
Ours		

Table 8: Qualitative comparison on nesting and cross-page table cases. **Left, `text` nested under `text`:** Ours recovers the implicit parent–child relation even though both nodes share the `text` category. Baselines more frequently flatten the relation or assign the child to the wrong parent. **Right, an 18-page table:** The table continues over 18 consecutive pages. Ours merges the complete sequence into one structure, whereas the competing models do not finish the full cross-page merge. Ground truth Prediction.

Qwen3.8		
GLM-5.3		
Kimi-K3		
DeepSeek-V4.1		
GPT-5.6-sol		

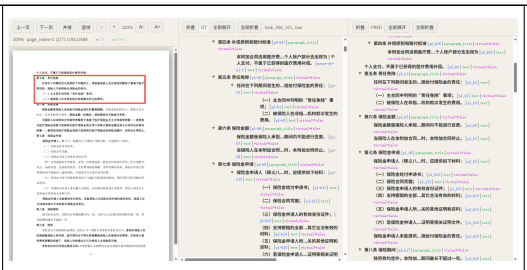
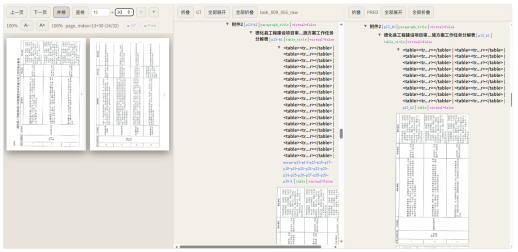
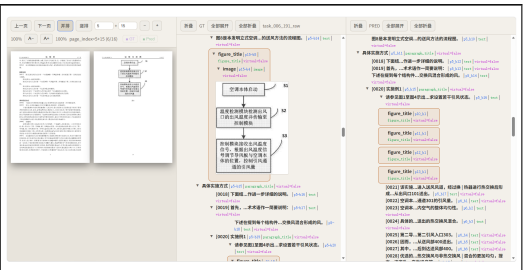
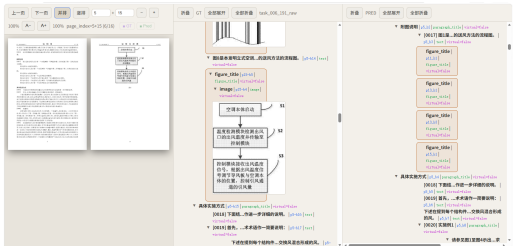
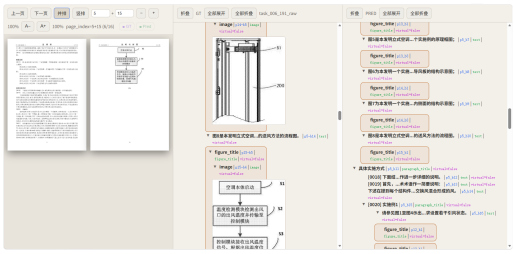
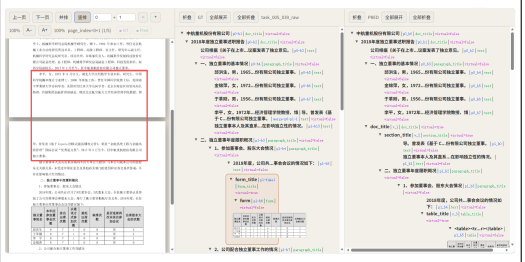
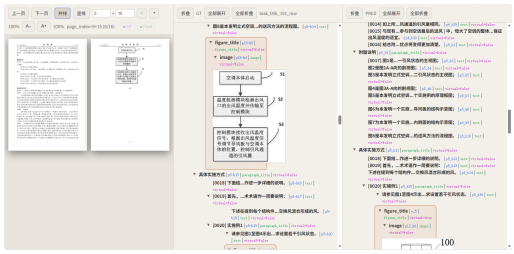
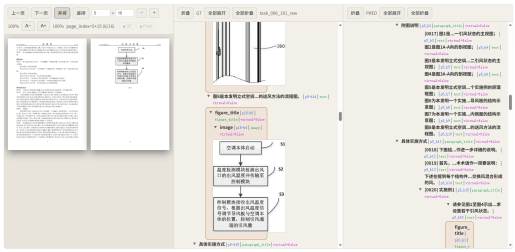
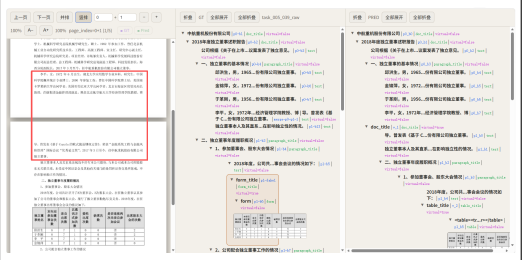
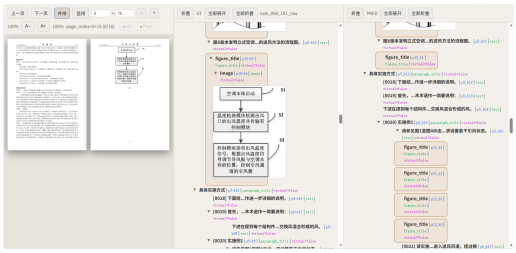
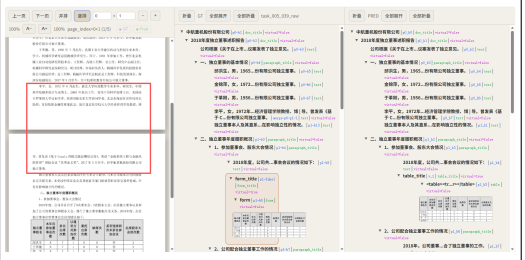
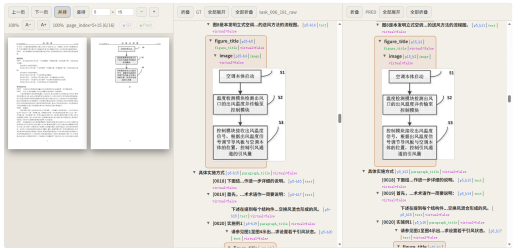
Claude-Fable-5		
Ours		

Table 9: Qualitative comparison on two long-range cases. **Left, cross-page paragraph:** Ours correctly continues the paragraph split by the page boundary. Baselines more often start a new node or attach the continuation to an incorrect branch. **Right, ten-page reference distance:** The textual anchor and visual target are separated by ten pages. Ours recovers the intended long-distance link, while competing methods more frequently miss the target or confuse it with another candidate. Ground truth Prediction.

Qwen3.8		
GLM-5.3		

Kimi-K3		
DeepSeek-V4.1		
GPT-5.6-sol		
Claude-Fable-5		
Ours		

B Dataset Statistics

Documents are assigned to training, development, and test by a stable hash of the document identifier, The realized split is 971, 155, and 91 documents.

B.1 DocVine Diversity Analysis

The distributions in Figure 4 demonstrate that the dataset objectively captures the composition and structural characteristics of real-world documents. In the upper-left panel, the dataset covers document types commonly encountered in real-world settings, moving document structure understanding closer to practical application scenarios; moreover, document type represents only a coarse dimension of categorization, since documents belonging to the same type can still exhibit substantial diversity in their layout and visual organization. In the upper-middle panel, the dominance of relations such as (paragraph_title,text), (text,text), and (paragraph_title,paragraph_title) shows that the dataset preserves the common hierarchical organization of headings, sections, and textual content found in actual documents. In the upper-right panel, the concentration of links from text to figure titles, references, and table titles reflects the way real documents use textual anchors to refer to visual elements, tabular information, and citations. In the lower-left panel, the predominance of within-page hierarchy edges, together with a substantial proportion of cross-page edges, demonstrates that the dataset captures both locally organized layouts and structural continuities across page boundaries. In the lower-middle panel, the near-exclusive concentration of merged text instances across two adjacent pages accurately reflects the typical continuation of paragraphs or text blocks caused by page breaks. In the lower-right panel, the broader page-span distribution of merged tables reflects the practical tendency of large tables to extend across multiple pages, thereby showing that the dataset retains complex cross-page structures found in real-world documents.



Figure 4: Overview of document composition and layout-structure statistics. From left to right, the top row presents the distribution of document types across 1,217 documents, hierarchy relation pairs across 124,680 edges, and reference relation pairs across 6,769 links. The bottom row shows the proportions of within-page and cross-page hierarchy edges, together with the page-span distributions of 1,975 merged text instances and 825 merged tables. Percentages are calculated within each chart, and relation pairs contributing less than 2% of the corresponding total are grouped as Other

B.2 Compare with Other Datasets

Table 10 places the corpus next to four published hierarchy datasets. Unlike arXivdocs and E-Periodica, which are single-page, our documents are multi-page: the longest has 298 pages, well beyond the maxima of HRDoc (35) and DocHieNet (50). The corpus has a moderate cross-page ratio (24.6%) but a long span (9.7).

As shown in Figure 5, DocVine is shorter in the bulk than DocHieNet and HRDoc: the median document has 4 pages (interquartile range 2–9), against medians of 9 and 11, and the upper whisker ends at 19 pages. arXivdocs and E-periodica are single-page. Maximum hierarchical depth is more comparable: DocVine has a median of 4 (interquartile range 3–6), with a whisker to 10 and a few documents as deep as 20. DocVine is therefore typically a moderately nested document, with a longer document tail than existing hierarchy datasets.

Table 10: Comparison with existing document-hierarchy datasets. M.P. is the maximum number of pages. C.P.R&S is the cross-page ratio and the maximum page span of cross-page hierarchical relations. A.M. is the annotation method. Rows other than DocVine are taken from prior work.

Dataset	Docs	Pages	M.P.	C.P.R&S	A.M.	Document type	Language
arXivdocs Rausch et al. (2021)	362	362	1	(0%, 0)	Manual	Scientific papers	En
HRDoc Ma et al. (2023)	2500	31651	35	(24.9%, 2.4)	Automatic	Scientific papers	En
E-Periodica Rausch et al. (2023)	542	542	1	(0%, 0)	Manual	Magazines	En, DE, FR, IT
DocHieNet Xing et al. (2024)	1673	15610	50	(37.4%, 5.4)	Manual	Multiple types	En, Zh
DocVine	1,217	12,736	298	(24.6%, 9.7)	Manual	Multiple types	Zh



Figure 5: Distribution of document length and maximum hierarchical depth. The gold line is the median, and the axes are broken so that long tails do not compress the boxes.

C Annotation Guidelines

C.1 Annotation Objective and the Two-Stage Boundary

The objective is not to reproduce OCR candidate boxes. It is to reconstruct a multi-page document as a computable and traceable semantic structure. The workflow is strictly divided into two stages.

Stage	Question addressed	Primary operations	Output
Stage I: Page-Level Annotation	Which semantic blocks exist on the current page, where are they located, and what are their categories?	Inspect each page; add, delete, merge, split, move, or resize bounding boxes; correct categories; and arrange blocks in the natural reading order.	Page-level annotation result
Stage II: Cross-Page Annotation	How do the blocks on all pages form a complete document hierarchy?	Map block references; build the forest; define parent–child relationships and node order; add merged members, virtual nodes, and `link_to` relationships.	Cross-page structure

The responsibilities of the two stages must remain separate:

- The page-level stage records only blocks that physically exist on a page. It does not create virtual nodes, establish parent–child relationships, perform cross-page merging, or mount reference relationships.
- The cross-page stage treats the approved page-level result as the sole source of real blocks and should not directly modify a bounding box or category.
- If a missing block, incorrect box, or incorrect category is discovered during cross-page annotation, the annotator must return to the page-level stage, correct the page-level result, and remap the affected block. A real block must not be fabricated in the forest.
- The page-level result expresses what is visibly present on a page. The cross-page result expresses where that content belongs in the document and how it is connected.

C.2 Stage I: Page-Level Annotation

C.2.1 Input, Output, and Completion Criteria

The input consists of page images arranged in the correct document order. Existing pre-annotations, including OCR output, may be used as references but must not be treated as final annotations.

The page-level stage is complete only when all of the following conditions are satisfied:

1. Every page has been inspected.
2. Every meaningful item is covered by a real block.
3. The boundary and category of every block are correct.

C.2.2 Page Preflight and Abnormal Samples

Before page-by-page annotation begins, the following conditions must be verified. If any condition cannot be satisfied, the affected page or document must be marked as abnormal and excluded from normal training samples:

- The page order is correct.
- Text and major visual objects are legible.

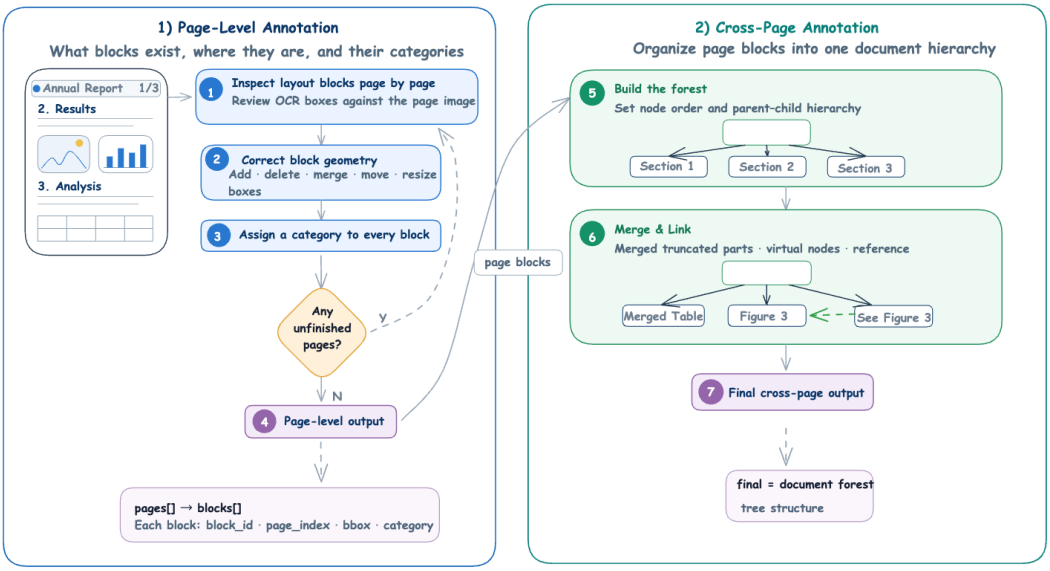


Figure 6: Two-stage workflow for document layout annotation. Page-level annotation refines block geometry, assigns categories and unique block identifiers, and produces `label.json`. Cross-page annotation organizes these blocks into a document hierarchy and adds merged members, virtual nodes, and `link_to` relations. Page-level attributes are then propagated to the final `multi-page.json` output.

- Web pages or electronic documents have been rendered completely.
- The page contains no severe overlap that makes semantic content or object boundaries unrecoverable.
- The primary reading direction and document type can be determined.

C.2.3 Page-by-Page Annotation Loop

Process each page in the following order:

1. Inspect all candidate blocks and all regions that may have been missed.
2. Delete noise, purely decorative content, repetitive logos, watermarks, and meaningless web controls.
3. Correct block geometry by adding, deleting, merging, splitting, moving, or resizing bounding boxes.
4. Assign an official category to every final block.
5. Arrange the blocks in the natural reading order of the page.
6. Check whether any page remains unfinished. If so, continue the page-by-page loop.

C.2.4 Fundamental Block-Definition Principle

One Box Represents One Category and One Semantic Unit A final block must satisfy both of the following conditions:

- All content within the box belongs to the same category.
- All content within the box forms one continuous semantic unit that cannot reasonably be divided further.

Content with different categories must be split. Content with the same category must also be split when it contains independent semantic units. Conversely, fragments created by pre-annotation should be merged when they belong to the same semantic unit.

C.2.5 PAGE-LEVEL CATEGORY SYSTEM

Real page blocks use the following official categories.

Category	Stored value	Page-level decision boundary
Body text	`text`	Ordinary paragraphs, explanatory text, table-of-contents entries, and list-item text; marginal notes also use this category.
Chart or diagram	`chart`	Flowcharts, infographics, statistical charts, and similar visual diagrams as a whole; internal text is not annotated separately.
Formula	`formula`	A formula together with an immediately adjacent equation number.
Document title	`doc_title`	The highest-level title of a document, article, news report, or independent subdocument.
Figure title	`figure_title`	The visible title or caption of an image, chart, or figure group.
Footer	`footer`	Repeated or fixed information at the bottom of a page, excluding footnote content.
Footnote	`footnote`	An in-page note or explanatory footnote associated with the main text.
Header	`header`	Repeated or fixed information at the top of a page, including elements such as a barcode on a patent page.
Image	`image`	A photograph, portrait, illustration, or other visual object that contributes meaningful content.
Paragraph title	`paragraph_title`	The title of a chapter, section, paragraph group, or clause.
Reference	`reference`	An individual bibliographic entry within a reference section.
Date	`date`	A standalone block whose primary semantic function is to express a date.
Seal	`seal`	An official seal, cross-page seal, or similar stamped object.
Signature	`signature`	A handwritten signature or signature mark.
Table	`table`	Data or form-like content with a row-and-column structure.
Table title	`table_title`	The visible title or caption of a table or table group.
Root	`root`	A system-generated document root; it is not annotated as a real page block.

C.2.6 NOISE AND OUT-OF-TREE ELEMENTS

Recognized units are cut into the three sets of §3. The two flags below are independent, and neither is implied by category alone.

`irrelevant` Set `irrelevant=true` only for noise: a running header, a footer, a page number, a table head restated on every page, or decoration. The unit is then dropped. It is not mounted in the forest, and it is not an endpoint of a reference link. The flag does not change the unit's category or bounding box.

`out_of_tree` Set `out_of_tree=true` for real content that is kept out of the hierarchy, such as a figure, a caption, or a bibliography entry that is not mounted in the forest. The same category may also stay in the tree. A figure, table, or bibliographic entry that belongs to the hierarchy keeps `out_of_tree=false` and is placed in V^+. An out-of-tree unit remains available as a reference

endpoint. It does not enter forest construction. The flag does not change the unit's category or bounding box.

If an older annotation file stores this decision in a field named `irrelevant`, that field means `out_of_tree` and not the noise flag above. New annotation uses `out_of_tree`.

C.3 Stage II: Cross-Page Annotation

C.3.1 Input and Node Mapping

The cross-page stage takes the page-level result that has passed quality inspection as its input.

C.3.2 Building the Forest

The forest represents the primary reading order and parent–child hierarchy of the document. Build it in the following sequence:

1. A system `root` is written for serialization. It is not a semantic node and not a member of V^+. Its children are the top-level elements of the document, and every element of V^+ reaches it.
2. Place every real unit of V^+ according to page order and within-page reading order. Units in V_{irr} are omitted. Units in V_{ext} are withheld from the forest and kept for reference linking.
3. Determine containment from top to bottom. A parent may be a title or any other real text block, such as a bolded lead phrase or the sentence that introduces a list.
4. Assign one direct parent to a unit that is subordinate to another unit. Leave the parent empty when the unit has no semantic parent in V^+. After the forest is built, hang every remaining root of V^+ from the system `root`. A footnote is an isolated node in V^+ and hangs from `root` once the tree is closed.
5. Confirm the order of siblings that share a parent.
6. Where a required parent has no corresponding text on the page, add a virtual node. Do not invent a virtual node for visible text the page already contains.

The forest must satisfy all of the following constraints:

- Every real unit in V^+ appears exactly once.
- Every node has at most one direct parent, and the parent link introduces no cycle.
- An isolated node has no parent in V^+. It hangs from the system `root` in the serialized tree.
- Siblings under the same parent follow the true reading order.
- The output order is a preorder traversal, so every parent appears before all of its descendants.
- The reading order of units on the same page is not changed without evidence.

C.3.3 Virtual Nodes

A virtual node is a parent that the structure requires and that has no corresponding text on the page. It carries neither a category, nor a box, nor text. It hosts elements already in V^+. Visible text that was missed by a detector is added as a real unit at page level. A bolded lead phrase or a list introducer that is real text on the page is a real parent, not a virtual node. The serialization `root` is not a virtual node.

Create a virtual node when a group needs a shared parent and the page has no text that can serve as that parent. This includes a document with no visible overall title, a group of paragraphs with no heading text, a list item split across peer paragraphs, and a figure or table group with no caption on the page. When usable text on the page can serve as the parent, use that real node. A virtual node is not a reference anchor. It may be the target of a reference link.

C.3.4 MERGED MEMBERS AND ATTACHMENT

`merge` The `merge` operation indicates that multiple real blocks jointly form one semantic node. It may be used only when direct continuity is supported by evidence:

- The same natural paragraph is interrupted by a page boundary and continues from the unfinished text on the next page.
- The same title or semantic unit is split by a page boundary.
- The same table continues directly on the next page without a new title, a new header, or an independent semantic starting point.

Merged nodes must satisfy the following requirements:

- The merged node records references to all real member blocks and preserves the page number and bounding box of each member.
- No merged member may also be mounted elsewhere in the main tree as an independent node.

`attach` The `attach` operation mounts an independent node or subtree under its semantically correct parent.

- If a new node is a peer of an existing node, attach it to their common parent.
- If a new node belongs to the content governed by a real or virtual parent, attach it to that parent.
- Attach a `table` to the corresponding `table_title`; attach an `image` or `chart` to the corresponding `figure_title`.
- If a new node has no semantic parent in V^+, leave the parent empty during construction. After the last merge round, hang it from the system `root`. A footnote remains an isolated node in V^+ and hangs from `root` once the tree is closed.
- A virtual node may serve as an attachment parent.

C.3.5 DEDUPLICATION OF REPEATED CROSS-PAGE ELEMENTS

Deduplication decides whether a current-page unit repeats chrome from the preceding page. It does not decide out-of-tree membership, and it does not merge a section continuation. A repeated header, footer, page number, or restated table head is noise and receives `irrelevant=true`. A continuation of the same paragraph, title, or table stays in V^+ and is handled by `merge`.

An element on the current page may be treated as repeated chrome only when all of the following conditions hold:

1. The categories are identical or semantically equivalent.
2. The texts are essentially identical after whitespace and punctuation are removed; minor OCR differences are acceptable.
3. Their relative positions within the pages are consistent or occur at approximately the same height.

C.3.6 LINK_TO REFERENCE RELATIONSHIPS

The `link_to` field serializes one reference link. It records a locatable anchor and its target. It does not move the target, change the target's parent, or change the target's membership. Several links may share one anchor unit. The anchor unit and the target are drawn from $V^+ \cup V_{\text{ext}} \cup V_{\text{v}}$. A target may be a leaf, a non-leaf entity, or a virtual node. A unit in V_{irr} is not an endpoint.

Create reference links in the following sequence:

1. Identify a locatable anchor. It may be explicit, as in "see Figure X," "as shown in Table X," "Reference [X]," or "see Section X for details." It may also be implicit, as in a superscript pointing at its footnote or a phrase such as "as discussed in the previous chapter."
2. Locate the target using its number, name, page, category, and layout position. The target may be a figure, table, section, appendix, citation, footnote, or other unit in $V^{+} \cup V_{\text{ext}} \cup V_{\text{v}}$.
3. Create `link_to` only when the evidence is sufficient; do not create a link when the specific target cannot be confirmed.
4. One anchor may refer to multiple targets, and one target may be referenced by multiple anchors.
5. A target that belongs in the hierarchy stays in the forest under its original parent. A target marked `out_of_tree` stays outside the forest. Figures, tables, and bibliographic entries that are part of the hierarchy are not marked out-of-tree merely because a citation exists.

D VinePipe Prompts

This appendix works out the input prompt and output schema of each inference task in full. The prompt, a concrete model response, and the operation behind every output entry are shown together, so each schema reads as a complete example rather than as isolated fields.

D.1 Semantic-Block Correction

D.1.1 Prompt and input

The correction task is prompted with a system message that states the task and its output contract, together with a user message that carries the page image and the candidate blocks produced by the OCR front-end.

```
[System message]
You are given a page image and a list of candidate blocks detected
by an OCR front-end. Correct them into semantic blocks ordered by
reading order. You may merge, split, or adjust the boundary of
candidate blocks, and you may add blocks for regions the front-end
missed. For every corrected block, either reuse a candidate box by
emitting its id as "source_id", or redraw the box by emitting a
"bbox". Assign each block a semantic "category". Mark noise such as a
running header, footer, page number, restated table head, or
decoration with "irrelevant": true. Mark real content that stays
out of the hierarchy with "out_of_tree": true.

[User message]
<page image>
{"id":"b1","bbox":[10,20,310,58],"text":"1. Introduction"}
{"id":"b2",
 "bbox":[10,70,310,150],
 "text":"Deep learning has ... 2. Method"}
{"id":"b3","bbox":[10,160,150,210],"text":"Table 1: Results"}
{"id":"b4","bbox":[160,160,310,210],"text":"0.92 0.87"}
{"id":"b5","bbox":[10,220,310,240],"text":"Figure 1: Overview"}
```

D.1.2 Response and operations

The model returns the corrected blocks in reading order. Each entry corresponds to one correction operation.

```
{"source_id":"b1","category":"paragraph_title"}
{"bbox":[10,68,310,110],"category":"text"}
{"bbox":[10,118,310,130],"category":"paragraph_title"}
{"bbox":[10,158,310,212],"category":"table"}
{"source_id":"b5","category":"figure_title","out_of_tree":true}
{"bbox":[10,300,310,330],"category":"footnote"}
```

The entries map to operations as follows.

- {"source_id":"b1","category":"paragraph_title"} *reuses* the box of b1 and *re-categorizes* it from text to paragraph_title. A candidate box whose boundary is already correct is referenced by source_id rather than redrawn.
- The two bbox entries at lines 2–3 *split* b2, which mixed a body paragraph with the heading "2. Method", into two blocks.
- The bbox entry at line 4 *merges* b3 and b4, the two halves of one table, into a single table block whose box covers both halves.
- {"source_id":"b5","category":"figure_title","out_of_tree":true} *reuses* b5, *re-categorizes* it from text, and *keeps this instance out of the hierarchy*. The unit remains real content and is eligible as a reference target. The same category stays in V^+ when that instance is mounted in the tree. Noise such as a running header is a separate flag, irrelevant=true, and is dropped.
- The final bbox entry *supplements* a footnote that the front-end missed entirely.

A pure boundary adjustment that neither splits nor merges takes the same bbox form. It differs only in that the redrawn box stays within a single candidate.

D.1.3 Choosing reuse versus redraw

Whether a ground-truth block reuses a candidate box or redraws its own is decided at training time by matching the block g against the candidate set. A candidate box b is a match when the intersection-over-self ratios on both sides each exceed a single threshold $\tau \in [0, 1)$:

$$\mathrm{match}(g, b) = \mathbb{I}\left[\frac{|g \cap b|}{|g|} \geq \tau \ \wedge \ \frac{|g \cap b|}{|b|} \geq \tau\right]. \tag{4}$$

A higher τ admits fewer candidate matches and forces more redraws. A lower τ admits more source_id reuse at the cost of looser box fidelity. The same τ is used for every training run and held fixed across all experiments reported in this paper.

D.2 Cross-Page Deduplication

D.2.1 Prompt and input

The deduplication task is prompted with a system message that states the task and its output contract, together with a user message that carries the corrected block lists of two adjacent pages and their images.

```
[System message]
You are given the corrected block lists of two adjacent pages.
Output the ids of current-page blocks that duplicate a block on
the preceding page. Decide from text content, semantic category,
and layout position together. Do not drop a real section
continuation on textual similarity alone.

[User message]
<page image 1> <page image 2>
preceding page:
{"id":"a1",
 "category":"title",
 "bbox":[90,40,520,70],
 "text":"Annual Report 2024"}
{"id":"a2",
 "category":"text",
 "bbox":[90,120,520,240],
 "text":"Revenue grew 12% year over year, ..."}
{"id":"a3",
 "category":"paragraph_title",
```

```
 "bbox":[90,260,200,280],
 "text":"2.1 Market"}
current page:
{"id":"c1",
 "category":"title",
 "bbox":[90,40,520,70],
 "text":"Annual Report 2024"}
{"id":"c2",
 "category":"text",
 "bbox":[90,90,520,220],
 "text":"Revenue grew 12% year over year, driven ..."}
{"id":"c3",
 "category":"text",
 "bbox":[90,240,520,320],
 "text":"by stronger demand in the Asia-Pacific region."}
```

D.2.2 Response and operations

The model returns the ids of the current-page blocks to ignore.

```
{"ignore_ids":["c1"]}
```

The entries map to operations as follows.

- `c1` duplicates `a1`. The two share identical text, the same `title` category, and the same top-of-page position, so `c1` is the per-page restated title and is dropped.
- `c2` and `c3` continue the `a2` sentence across the page seam. Their text overlaps `a2`, but their categories and positions mark them as a real section continuation, so they are kept.

D.3 Per-Page Hierarchical Trees

D.3.1 Prompt and input

The tree task is prompted with a system message stating the output contract, together with the corrected and deduplicated block list of a single page and its image.

```
[System message]
You are given a single page image and its corrected, deduplicated
block list. Output a preorder node sequence. Every node has a
"parent_id", which is null for a forest root or an isolated node.
You may generate a virtual parent, with "is_virtual": true, only
when the structure requires a parent and the page has no text
that can serve as that parent. A virtual node has neither a
category, nor a box, nor text.

[User message]
<page image>
{"id":"n1",
 "bbox":[90,90,520,150],
 "text":"We first describe the data.",
 "category":"text"}
{"id":"n2",
 "bbox":[90,170,520,280],
 "text":"Second, we train the model ...",
 "category":"text"}
```

D.3.2 Response and operations

The model returns a preorder sequence. Virtual nodes are emitted first so that their descendants can reference them.

```
{"id":"v_1",
 "is_virtual":true,
```

```
 "parent_id":null}
{"id":"n1","parent_id":"v_1"}
{"id":"n2","parent_id":"v_1"}
```

The entries map to operations as follows.

- `v_1` is a virtual parent. The two paragraphs need a shared parent, and the page has no text that can serve as that parent. The node has neither a category, nor a box, nor text. Visible text missed by the detector would instead be added as a real unit during semantic-block correction.
- `n1` and `n2` attach to `v_1`. Neither real block is modified.

D.4 Progressive Cross-Page Merging

D.4.1 Prompt and input

The merging task receives a left tree group and a right tree group. The left side contributes its active structure with a `path_idx` on every actionable node, and the right side contributes its top-level nodes. Two boundary images at the page seam are optional.

```
[System message]
You are given a left tree and the top-level nodes of a right tree.
For every top-level node on the right, choose "merge", "attach",
or "keep". "merge" continues an existing left node across the page
and needs a "merge_path_idx". "attach" places the node under a
left parent and needs a "parent_path_idx". "keep" leaves the parent
empty, for a forest root or an isolated node, and carries no index.

[User message]
<page image 1> <page image 2>
left tree (active structure):
{"path_idx":0,
 "id":"t1",
 "category":"title",
 "text":"Annual Report 2024"}
{"path_idx":1,
 "id":"t2",
 "category":"paragraph_title",
 "text":"2. Financial Review"}
{"path_idx":2,"id":"t3","category":"table","text":"Table 3: Revenue"}
right tree (top-level nodes):
{"node_idx":0,
 "id":"r1",
 "category":"table",
 "text":"Table 3: Revenue (cont.)"}
{"node_idx":1,
 "id":"r2",
 "category":"paragraph_title",
 "text":"2.1 Revenue by Region"}
```

D.4.2 Response and operations

The model emits one action per top-level node, in `node_idx` order.

```
{"node_idx":0,"action":"merge","merge_path_idx":2}
{"node_idx":1,"action":"attach","parent_path_idx":1}
```

The entries map to operations as follows.

- `r1` continues the `Table 3: Revenue` node on the left. The model emits `merge` with `merge_path_idx`:2, so the members, text, and boxes of `r1` are concatenated onto `t3` in order-preserving fashion.

- `r2` starts a new subsection. The model emits `attach` with `parent_path_idx`:1, so `r2` is appended under the section `t2`.

D.5 Reference Construction

D.5.1 Prompt and input

Anchors and targets are partitioned into fixed-size groups, and a Cartesian product of anchor and target groups forms each sample. Every sample carries candidate-node ids, categories, pages, coordinates, text summaries, and the relevant page images.

```
[System message]
You are given a group of reference anchors and a group of candidate
targets. For every anchor, link it to the target it refers to.
Emit only high-confidence "anchor_id"--"target_id" pairs.

[User message]
<page images>
anchors:
{"anchor_id":"a1",
 "category":"text",
 "text":"... as shown in Table 2 ..."}
{"anchor_id":"a2","category":"text","text":"... see Figure 1 ..."}
targets:
{"target_id":"t1",
 "category":"figure_title",
 "text":"Figure 1: Overview"}
{"target_id":"t2","category":"table_title","text":"Table 2: Results"}
{"target_id":"t3",
 "category":"table_title",
 "text":"Table 5: Ablation"}
```

D.5.2 Response and operations

The model returns only the pairs it is confident about.

```
{"pairs":[{"anchor_id":"a1","target_id":"t2"},
        {"anchor_id":"a2","target_id":"t1"}]}
```

The entries map to operations as follows.

- `a1` ("as shown in Table 2") links to `t2` ("Table 2: Results").
- `a2` ("see Figure 1") links to `t1` ("Figure 1: Overview").
- `t3` receives no link. Anchors without a confident target are simply left unpaired.

E MPDocBench-Parse by Category

Table 13 is the category breakdown summarized in §5.3.

F Inference and Post-Processing

This appendix expands Section 4's pipeline. It first analyzes the multi-round reduction scheme used by progressive cross-page merging. It then spells out the deterministic post-processing rules that materialize per-page tree predictions, execute merge and attach actions across page boundaries, and integrate long-range reference pairs.

Table 13: Title Hierarchy Recovery Performance across Document Types

Model	Overall	Annual Report	Book	Financial Report	Exam Paper	Government Document	National Standard	Handwritten Document	Magazine	User Manual	Newspaper	Academic Paper	PPT Slide	Textbook	Research Report	Prospectus
Pipeline-Based Specialized Vision-Language Models																
GLM-OCR Duan et al. (2026)	44.79	46.25	43.26	46.49	67.26	57.09	27.93	50.21	22.68	40.62	42.01	68.76	31.28	26.12	46.19	36.07
PaddleOCR-VL-1.5 Cui et al. (2026)	46.78	43.05	44.48	64.26	65.20	51.16	41.22	47.53	26.17	45.08	37.06	77.50	36.47	31.68	42.30	41.42
MinerU2.5 Niu et al. (2025)	36.90	32.82	31.73	26.53	60.75	63.81	23.94	50.53	20.20	36.51	24.28	57.08	29.81	18.87	37.79	20.40
Youtu-Parsing Yin et al. (2026)	44.28	42.85	42.19	37.13	65.99	73.75	41.86	47.44	28.93	46.47	25.20	45.12	38.08	26.87	42.16	42.61
MonkeyOCR-pro-3B Zhang et al. (2025)	40.34	39.10	36.31	30.64	56.59	60.78	20.03	50.80	23.82	41.60	30.55	58.68	31.16	25.06	43.49	35.49
Dolphin-v2 Feng et al. (2026)	50.04	52.99	48.91	39.01	62.25	68.91	41.28	48.79	28.54	43.70	42.41	76.41	45.59	31.93	52.05	45.84
End-to-End Specialized Vision-Language Models																
dots.mocr Zheng et al. (2026)	33.68	40.48	35.32	33.02	49.41	45.43	24.85	27.41	16.85	36.56	5.97	55.06	10.82	25.06	37.49	31.05
FireRed-OCR Wu et al. (2026)	50.84	44.06	46.70	41.27	71.96	74.45	37.47	48.14	28.25	49.55	44.02	75.94	32.39	35.51	56.41	48.64
dots.ocr Li et al. (2025)	45.12	50.63	39.69	45.86	56.62	66.94	52.74	40.70	28.06	42.78	26.83	57.96	34.22	29.07	52.15	41.00
DeepSeek-OCR2 Wei et al. (2026)	49.53	47.76	38.43	42.75	71.99	71.95	36.33	48.65	26.80	43.50	42.36	68.62	33.05	36.46	53.71	57.09
OCRVerse Zhong et al. (2026)	35.45	30.33	25.89	34.74	56.38	40.86	36.64	44.97	18.20	27.40	19.09	56.58	22.18	27.30	46.71	29.43
Logics-Parsing-v2 Chen et al. (2025b)	34.62	29.70	28.98	24.84	58.04	60.89	20.85	28.26	16.27	36.24	25.07	56.35	32.30	19.62	32.27	26.06
Qianfan-OCR Dong et al. (2026)	49.19	46.40	41.44	42.60	67.24	66.25	32.50	53.64	26.88	45.20	23.30	75.09	35.85	36.86	61.81	53.23
ChandraOCR 2 Datalab (2025)	48.64	48.94	42.44	42.46	67.34	71.18	47.60	56.08	27.40	52.29	27.46	61.00	45.49	31.53	52.29	44.37
General-Purpose Vision-Language Models																
Gemini-3.1-pro-preview Google DeepMind (2026)	26.94	20.41	23.11	21.02	50.54	36.61	14.95	33.72	25.71	26.55	24.67	18.62	34.99	29.21	18.88	20.66
ChatGPT-5.2-2025-12-11 OpenAI (2025)	37.50	41.34	38.89	46.49	40.57	50.03	34.48	44.34	20.79	37.00	23.52	56.55	34.95	24.54	36.82	28.77
Qwen3.6-plus TongYi (2025)	32.06	31.58	31.17	19.73	55.15	49.40	22.71	31.08	20.41	24.23	17.69	29.31	45.86	31.07	25.90	31.33
Qwen3-VL-235B Bai et al. (2025)	42.34	49.71	37.14	50.85	59.23	58.60	42.23	43.36	22.08	42.05	32.22	60.03	42.15	25.29	35.53	30.57
InternVL-3.5-38B Wang et al. (2025)	27.01	24.25	26.42	31.95	54.91	49.81	11.78	34.68	14.36	25.25	11.88	31.93	32.04	13.41	19.33	15.76
VineVLM-2B	55.30	63.87	50.34	81.54	56.93	58.99	50.28	54.81	31.24	57.36	34.38	81.93	55.85	42.61	46.04	61.98

F.1 Multi-Round Reduction Complexity

For a document of P pages, progressive cross-page merging performs $P-1$ tree-pair operations. The approximate number of sequential dependency rounds is

$$R = \lceil \log_2 P \rceil. \tag{5}$$

The $P-1$ pairs are spread across $\lceil \log_2 P \rceil$ sequential rounds. Multiple pairs in the same round can be inferred in parallel, so the wall-clock depth is $O(\log_2 P)$ even though the total work is $O(P)$. Pairwise reduction does not decrease the total number of actions, but it controls the tree size in each input and enables parallel inference within a round.

F.2 Per-Page Tree Materialization

The per-page tree model predicts only the node order, `parent_id`, virtual-node information, and reference-anchor indicators. The text, category, coordinates, and page information of real nodes are restored from the corrected semantic blocks. This design separates hierarchical prediction from node-content generation. It prevents the model from modifying or omitting previously determined page content. The model output is a preorder sequence

$$\hat{\mathcal{S}}_p = [(n_i,\ \mathrm{parent}(n_i))]_{i=1}^{N_p}, \tag{6}$$

where n_i is the i-th node on page p and $\mathrm{parent}(n_i)$ is its predicted parent.

During post-processing the system maintains an active ancestor path according to the node order and inserts each node into the `children` list of its predicted parent. The output sequence is materialized as a valid per-page tree only if the following constraints are satisfied:

$$\forall n_i, \qquad n_i \text{ appears exactly once in } \hat{\mathcal{S}}_p, \tag{7}$$

$$\mathrm{parent}(n_i) \neq \varnothing \implies \mathrm{parent}(n_i) \in \{n_1, \ldots, n_{i-1}\}, \tag{8}$$

and the predicted parent remains on the active preorder traversal path. The set and order of all real nodes in the output must exactly match those of the input semantic blocks. This prevents real nodes from being deleted, duplicated, or reordered during tree generation.

F.3 Cross-Page Action Execution and Reference Integration

The cross-page merging model does not regenerate the complete document tree. Instead, it predicts a `merge` or `attach` action for each top-level subtree on the right-hand side. Before executing these actions, the post-processing module builds preorder path indices from the active branches of the left-hand document tree and verifies that the action list covers every incoming top-level node exactly once.

For a `merge` action, the incoming node is regarded as a continuation of an existing node across a page boundary. Its members, page indices, categories, bounding boxes, text, and child nodes are concatenated in order-preserving fashion with those of the target. Given a target node u and an incoming node v, the operation is

$$\mathrm{merge}(u, v): \quad u \leftarrow u \| v, \tag{9}$$

where $\|$ denotes order-preserving concatenation across all five ordered attributes (`member`, `page`, `bbox`, `text`, `children`). For an `attach` action, the complete incoming subtree is retained and appended to the `children` list of the specified parent node. After each merging round, the active path indices are recomputed for the next round. Once all pages have been merged, globally unique virtual-node identifiers are reassigned according to the preorder traversal of the final document tree.

During reference linking, the post-processing module verifies that each predicted anchor and target belongs to the corresponding candidate group; duplicate anchor–target pairs and self-references are removed. For every valid reference pair, the `link_to` field of the anchor records the edge. A target in the forest keeps its parent. A target in V_{ext} stays outside the forest. The field does not change membership.

The final output therefore contains two complementary types of relationships. The first is parent–child relationships represented by the `children` field. The second is long-range semantic references represented by the `link_to` field. Together they form a structured document representation. The representation can be directly used by downstream retrieval, analysis, and document-understanding applications.

G Training Objective

VinePipe is trained as one vision language model. Samples from its stages are rendered in a unified multi-turn message format. Let $\mathcal{T}$ be the task set, let N_t be the number of samples for task t, and let N be the total number of samples. The overall training objective is

$$\mathcal{L} = \sum_{t \in \mathcal{T}} \lambda_t \sum_{i=1}^{N_t} \mathcal{L}_{\text{CE}}\big(y_i^t,\ f_\theta(x_i^t)\big)\,. \tag{10}$$

To alleviate the sample-count imbalance between page-level and document-level tasks, inverse-frequency task weights are computed from the number of training samples:

$$\lambda_t = \text{clip}\left(\frac{N}{|\mathcal{T}|N_t},\ \lambda_{\min},\ \lambda_{\max}\right). \tag{11}$$

The weights are clipped to $[\lambda_{\min}, \lambda_{\max}]$ to prevent any single task from being either fully suppressed or fully dominating the gradient. The same recipe is held fixed across every training run reported in this paper.

H Evaluation Metrics: Protocol

H.1 Evaluation objects and shared spatial matching

Let T and $\widehat{T}$ be the ground-truth and predicted ordered forests. Each contains real nodes V_{r} and virtual nodes V_{v}. A fixed system root connects all top-level nodes. Sibling order is the order submitted to the evaluator. Reference relations are kept separate from the hierarchy. Hierarchy scoring traverses the tree starting from the system root; target subtrees stored only under `link_to` are outside that traversal. Every node, including every virtual node, must have an ID unique within its own tree; equality of raw IDs across trees is not matching evidence.

For a real node v, let $\mathcal{B}_p(v)$ contain its rectangles on page p, and define

$$U_p(v) = \bigcup_{b \in \mathcal{B}_p(v)} b, \qquad I(u,v) = \frac{\sum_p |U_p(u) \cap U_p(v)|}{\sum_p |U_p(u) \cup U_p(v)|}. \tag{12}$$

An absent page contributes an empty region. The implementation returns zero for zero union area. Overlapping rectangles are counted once; different pages never intersect. Because a single node may contain multiple bounding boxes on the same page (e.g., split paragraphs or multi-column text), this metric is defined as a ratio of accumulated areas rather than an average of rectangle-wise or page-wise IoUs.

Let $\gamma \in \{0, 1\}$ be the category-matching option and $c(v)$ the category normalized to a sorted set of strings. Define compatibility as

$$C_\gamma(u, v) = \begin{cases} 1, & \gamma = 0, \\ [c(u) = c(v)], & \gamma = 1. \end{cases} \tag{13}$$

The default is $\gamma = 0$ to bypass potential category vocabulary mismatches across datasets.

The candidate set for real-node alignment is

$$\mathcal{C}_{\tau,\gamma} = \{(u, \widehat{v}) \in V_{\mathrm{r}}(T) \times V_{\mathrm{r}}(\widehat{T}) : I(u, \widehat{v}) \geq \tau,\ C_\gamma(u, \widehat{v}) = 1\}. \tag{14}$$

We choose a one-to-one matching $M_{\tau,\gamma} \subseteq \mathcal{C}_{\tau,\gamma}$ that lexicographically maximizes

$$\left(|M_{\tau,\gamma}|, \sum_{(u,\widehat{v}) \in M_{\tau,\gamma}} I(u, \widehat{v}) \right). \tag{15}$$

Thus, maximizing the number of admissible correspondences precedes maximizing their total IoU. Hierarchical correctness is not used to choose this real-node alignment. The resulting map $\phi(\widehat{v}) = u$ is used by Hierarchy Edge F1. The IoU matrix is computed once and is also supplied to Soft-TEDS.

H.2 Soft-TEDS on one complete ordered tree

For two ordinary nodes, the substitution cost is

$$c(u, \widehat{v}) = \begin{cases} 1, & \text{exactly one node is virtual,} \\ 1 - C_\gamma(u, \widehat{v}), & \text{both nodes are virtual,} \\ 1, & \text{both are real and } C_\gamma(u, \widehat{v}) = 0, \\ \delta(u, \widehat{v}), & \text{both are real and compatible,} \end{cases} \tag{16}$$

with

$$\delta(u, \widehat{v}) = 1 - I(u, \widehat{v}). \tag{17}$$

All ordinary-node insertions and deletions cost one, including virtual nodes. The system roots are fixed to each other and cannot participate in other edits. The ordered tree-edit algorithm minimizes the total cost under ancestor and left-to-right order constraints.

A zero virtual-to-virtual substitution cost means only that their labels agree under the chosen category policy. It does not make their subtrees interchangeable at zero cost. Missing groups, changed nesting, and reordered distinguishable descendants are still evaluated by the full tree edit. In particular, exchanging two virtual sibling subtrees reverses the order of their real descendants and generally incurs a positive edit cost. Exchanging structurally and spatially indistinguishable subtrees need not change the score.

$$S(T, \widehat{T}) = clip\left(1 - \frac{D(T, \widehat{T})}{\max(|V(T)|, |V(\widehat{T})|)}, 0, 1\right), \tag{18}$$

Equation 18 is applied with $|V| = |V_{\mathrm{r}}| + |V_{\mathrm{v}}|$. When exactly one tree is empty, its score is zero. When both are valid empty trees, its score is one. A missing or malformed prediction is a failure, not a valid empty-tree special case.

H.3 Hierarchy Edge F1 through anonymous virtual paths

For each real node v, let $a(v)$ be its nearest strict real ancestor, or system root when none exists. Let

$$s_\gamma(v) = (q_1, \ldots, q_k)$$

encode the intervening virtual nodes, ordered from the ancestor toward v. When $\gamma = 0$, every q_i is the same marker $\mathtt{V}$; when $\gamma = 1$, q_i is the normalized category of that virtual node. A direct real-parent attachment has $k = 0$.The ground-truth record set is

$$\mathcal{P}(T) = \{(a(v), s_\gamma(v), v) : v \in V_{\mathrm{r}}(T)\}. \tag{19}$$

The predicted set $\widehat{\mathcal{P}}$ is constructed identically, with both real endpoints mapped through ϕ. The system root has a fixed shared identity. Unmatched real endpoints remain explicit. Virtual nodes produce no standalone records and require no one-to-one virtual-node matching.

Set intersection and differences give

$$TP = |\mathcal{P}(T) \cap \widehat{\mathcal{P}}|, \qquad FP = |\widehat{\mathcal{P}} \setminus \mathcal{P}(T)|, \qquad FN = |\mathcal{P}(T) \setminus \widehat{\mathcal{P}}|. \tag{20}$$

We report

$$P = \frac{TP}{TP + FP}, \qquad R = \frac{TP}{TP + FN}, \qquad F_1 = \frac{2TP}{2TP + FP + FN}. \tag{21}$$

Two valid empty record sets have $P = R = F_1 = 1$; otherwise a zero-denominator precision or recall is zero. Paths from the system root to real top-level nodes are included. Excluding the root from the TEDS node count does not remove these path records.

H.4 Link F1: implemented alignment and reporting scope

Link F1 evaluates directed anchor–target pairs, separately from hierarchical paths. In the current end-to-end evaluator, real anchors use ϕ. Each predicted bbox-bearing target is independently matched to the ground-truth candidate with maximum page-aware IoU, subject to the same threshold τ. Virtual target containers are recursively expanded to their real children.

This expansion defines Link F1 as recovery of the referenced real elements. Virtual-target IDs, nesting, and categories do not enter the relation records. This metric measures reference resolution, not target hierarchy.

Let $\mathcal{L}$ be the ground-truth pair set, $\widehat{\mathcal{L}}_{\text{mapped}}$ the mapped prediction pair set, and U the number of extracted predictions whose anchor or target cannot be mapped. Then

$$TP_{\text{L}} = |\mathcal{L} \cap \widehat{\mathcal{L}}_{\text{mapped}}|, \tag{22}$$

$$FP_{\text{L}} = |\widehat{\mathcal{L}}_{\text{mapped}} \setminus \mathcal{L}| + U, \tag{23}$$

$$FN_{\text{L}} = |\mathcal{L} \setminus \widehat{\mathcal{L}}_{\text{mapped}}|. \tag{24}$$

Equation 21 gives Link F1, with the same empty-set convention. Link precision, recall, and F1 are macro-averaged over the same fixed document list as the three hierarchy scores.

H.5 Merge F1: cross-page fragment pairs

Merge F1 evaluates identity across pages, separately from hierarchical paths and from reference links. It measures whether page-level fragments of one real node are recovered as one identity; it does not score parent–child structure.

For a real node v, write v_p for its fragment on page p, that is the region $U_p(v)$ when that region is nonempty. The ground-truth merge-pair set is the set of unordered pairs of fragments that share a node and lie on different pages,

$$\mathcal{M}(T) = \big\{\{v_p, v_q\} : v \in V_{\text{r}}(T),\ U_p(v) \neq \emptyset,\ U_q(v) \neq \emptyset,\ p < q\big\}. \tag{25}$$

A node that spans three pages therefore contributes three pairs. The predicted set $\widehat{\mathcal{M}}$ is constructed identically from $\widehat{T}$.

Fragments are aligned by the page-aware IoU of Equation 12 at threshold $\tau = 0.85$. A one-to-one matching ψ maps predicted fragments to ground-truth fragments. A predicted pair $\{\widehat{a}, \widehat{b}\}$ is a true positive only when both endpoints lie in the domain of ψ and $\{\psi(\widehat{a}), \psi(\widehat{b})\} \in \mathcal{M}(T)$. Unmatched endpoints remain distinct identities: any predicted pair that uses an unmatched fragment cannot enter the intersection and is counted as a false positive.

Let $\widehat{\mathcal{M}}_{\text{mapped}}$ be the set of predicted pairs whose both endpoints are matched by ψ. Then

$$TP_{\text{M}} = |\mathcal{M}(T) \cap \widehat{\mathcal{M}}_{\text{mapped}}|, \tag{26}$$

$$FP_{\text{M}} = |\widehat{\mathcal{M}}| - TP_{\text{M}}, \tag{27}$$

$$FN_{\text{M}} = |\mathcal{M}(T) \setminus \widehat{\mathcal{M}}_{\text{mapped}}|. \tag{28}$$

Table 14: Structural capabilities of the evaluated methods. VineVLM-2B supports the complete prediction pipeline, whereas each baseline exposes only a subset of the document structure.

Method	Page-level elements	Irrelevant-element removal	Heading hierarchy	Paragraph hierarchy	Cross-page merge	Same-page image–text link	Long-distance image–text link	Virtual nodes
DOSA Li et al. (2026)			✓	✓		✓		
DocParser Rausch et al. (2021)		✓	✓			✓		
Docling-Granite Livathinos et al. (2025)		✓	✓			✓		
TextIn TextIn (2025)			✓		✓			
MinerU-Popo Xu et al. (2026a)		✓	✓		✓	✓		
VineVLM-2B	✓	✓	✓	✓	✓	✓	✓	✓

Equation 21 gives precision, recall, and F1. A document is excluded from the macro average when both $\mathcal{M}(T)$ and $\widehat{\mathcal{M}}$ are empty. A missing or malformed prediction is scored 0 and remains in the denominator; it is not treated as a valid empty pair set.

I Capability-aligned Method

Capability-aware comparison. The evaluated methods expose substantially different levels of document structure. As summarized in Table 14, some methods only recover heading hierarchies, whereas others additionally model paragraph structure, cross-page merging, image–text associations, or virtual nodes. A checkmark indicates that the corresponding capability is explicitly represented by the method and is available for evaluation; an empty cell indicates that the capability is not provided.

These differences make a direct comparison of the raw output trees inappropriate. We therefore evaluate each baseline within the structural space that it can represent and process the ground truth and VineVLM-2B predictions under the same constraints.