MoDora: Tree-Based Semi-Structured Document Analysis System

BANGRUI XU, Shanghai Jiao Tong University, China
QIHANG YAO, Shanghai Jiao Tong University, China
ZIRUI TANG, Shanghai Jiao Tong University, China
XUANHE ZHOU*, Shanghai Jiao Tong University, China
YEYE HE*, Microsoft Research, United States
SHIHAN YU, Beihang University, China
QIANQIAN XU, Beihang University, China
BIN WANG, Shanghai AI Lab, China
GUOLIANG LI, Tsinghua University, China
CONGHUI HE, Shanghai AI Lab, China
FAN WU, Shanghai Jiao Tong University, China

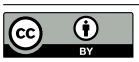
Semi-structured documents integrate diverse interleaved data elements (e.g., tables, charts, hierarchical paragraphs) arranged in various and often irregular layouts. These documents are widely observed across domains and account for a large portion of real-world data. However, existing methods struggle to support natural language question answering over these documents due to three main technical challenges: (1) The elements extracted by techniques like OCR are often fragmented and stripped of their original semantic context, making them inadequate for analysis. (2) Existing approaches lack effective representations to capture hierarchical structures within documents (e.g., associating tables with nested chapter titles) and to preserve layout-specific distinctions (e.g., differentiating sidebars from main content). (3) Answering questions often requires retrieving and aligning relevant information scattered across multiple regions or pages, such as linking a descriptive paragraph to table cells located elsewhere in the document.

To address these issues, we propose `MoDora`, an LLM-powered system for semi-structured document analysis. First, we adopt a local-alignment aggregation strategy to convert OCR-parsed elements into layout-aware components, and conduct type-specific information extraction for components with hierarchical titles or non-text elements. Second, we design the Component-Correlation Tree (CCTree) to hierarchically organize components, explicitly modeling inter-component relations and layout distinctions through a bottom-up cascade summarization process. Finally, we propose a question-type-aware retrieval strategy that supports (1) layout-based grid partitioning for location-based retrieval and (2) LLM-guided pruning for semantic-based retrieval. Experiments show `MoDora` outperforms baselines by 5.97%-61.07% in accuracy. The code is at *https://github.com/weAIDB/MoDora*.

CCS Concepts: • **Information systems** → **Retrieval tasks and goals**;

*Xuanhe Zhou and Yeye He are the corresponding authors.

Authors' Contact Information: Bangrui Xu, Shanghai Jiao Tong University, China, dreameternal@sjtu.edu.cn; Qihang Yao, Shanghai Jiao Tong University, China, yaoqihang@sjtu.edu.cn; Zirui Tang, Shanghai Jiao Tong University, China, tangzirui@sjtu.edu.cn; Xuanhe Zhou, Shanghai Jiao Tong University, China, zhouxuanhe@sjtu.edu.cn; Yeye He*, Microsoft Research, United States, yeyehe@microsoft.com; Shihan Yu, Beihang University, China, yushihan070611@qq.com; Qianqian Xu, Beihang University, China, 24421025@buaa.edu.cn; Bin Wang, Shanghai AI Lab, China, wangbin@pjlab.org.cn; Guoliang Li, Tsinghua University, China, liguoliang@tsinghua.edu.cn; Conghui He, Shanghai AI Lab, China, heconghui@pjlab.org.cn; Fan Wu, Shanghai Jiao Tong University, China, fwu@cs.sjtu.edu.cn.



ACM 2836-6573/2026/6-ART212
https://doi.org/10.1145/3802089

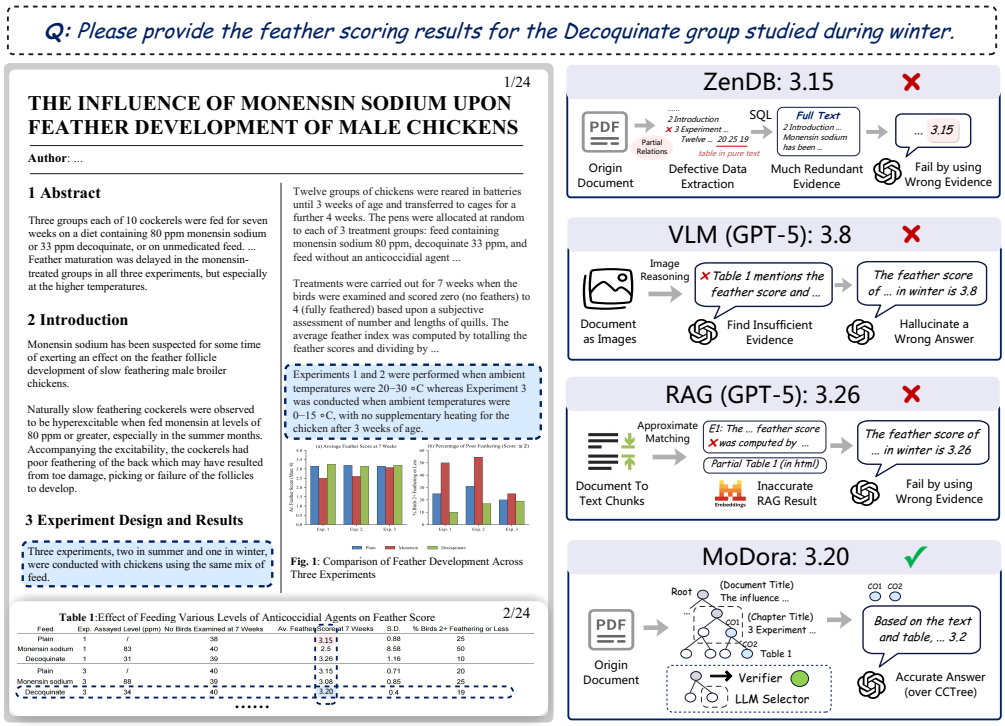


Fig. 1. Example Semi-Structured Document Analysis.

Additional Key Words and Phrases: Semi-Structured Document, Tree Structure, Information Retrieval

ACM Reference Format:
Bangrui Xu, Qihang Yao, Zirui Tang, Xuanhe Zhou, Yeye He*, Shihan Yu, Qianqian Xu, Bin Wang, Guoliang Li, Conghui He, and Fan Wu. 2026. MoDora: Tree-Based Semi-Structured Document Analysis System. *Proc. ACM Manag. Data* 4, 3 (SIGMOD), Article 212 (June 2026), 26 pages. https://doi.org/10.1145/3802089

1 Introduction

Semi-structured documents contain a mixture of diverse elements, such as tables, charts, free-form text, arranged in complex and often irregular layouts. These documents are widely observed in real-world scenarios, including scientific reports, financial statements, and technical manuals [9].

Example 1.1. Figure 1 presents a typical semi-structured scientific report containing components (COs) like (1) a data table of experimental results, (2) a chart visualizing trends from the table, and (3) accompanying text describing the setup and findings. While the table and chart offer structured insights, accurate question answering requires holistic interpretation across all components. Additionally, the report's hierarchical titles reflect the nested structures commonly found in semi-structured formats like HTML and JSON [19].

Based on our analysis of over one million real-world documents[1], we find that over 77% of the data contain at least one table, chart, or paragraph title. Among these, 61% include at least one

[1]Collected from representative sources like Scihub [2] and PDF drive [1] ([artifact]).

table and 40% contain at least one chart, demonstrating the ubiquity of documents with structured elements interleaved with free-form text.

Semi-structured document analysis aims to support natural language (NL) question answering over such documents, which is common and practically valuable. For the example in Figure 1, to answer the question (***Q***) about the feather state of specific group in winter, a human analyst needs to first infer from paragraphs on page 1 that Experiment 3 was conducted in winter, and then locates the corresponding feather score from the table on page 2.

Existing relevant methods can be broadly categorized into five classes, each exhibiting critical limitations in semi-structured document analysis [8, 9, 12, 13, 16–18, 20–22, 24–26, 28, 33]:

(1) Content extraction methods such as QUEST [26] and EVAPORATE [9] extract information (e.g., attribute–value pairs) and store it in relational tables for structured retrieval. While efficient for text content extraction, they discard document structure information, making it difficult to answer questions that depend on layout cues.

(2) Structure extraction methods such as ZenDB [20], DocAgent [25], SuperRAG [33], and PDFTriage [24] can extract hierarchical document structures (e.g., chapters and page mappings). However, they suffer from two main limitations. First, these methods typically represent documents at the page level, lacking fine-grained modeling. For instance, DocAgent does not explicitly model region-level coordinates within pages, which hinders precise location-based retrieval. Similarly, ZenDB fails to capture the structural and semantic relations between non-textual elements (e.g., tables, charts) and surrounding text, resulting in fragmented tree representations that limit downstream analysis. Second, the structural modeling methods lack robustness. For instance, ZenDB relies on visual feature clustering and LLM to detect titles and their body text. However, such clustering is error-prone, especially for complex document layouts, and can cause incorrect or flattened hierarchies.

(3) Programmatic methods like Palimpzest [21] require manually written domain-specific language (DSL) programs to operate over documents. This limits applicability in natural-language QA scenarios where no programs are provided.

(4) End-to-End models including pre-trained models (e.g., LayoutLMv3 [18], DocFormer [8], UDOP [28]) and Multimodal Large Language Models (MLLMs), such as DocOwl2 [17] and GPT-5 [22], process documents as images and answer in an end-to-end manner. However, these models often miss fine-grained textual details and above layout-specific relations, causing incomplete evidence aggregation.

(5) Retrieval-Augmented Generation (RAG) methods [12, 13, 16, 33] retrieve content through embedding and lack structural alignment, resulting in missing or misaligned evidence during multi-element reasoning. TextRAG retrieves relevant semantic pieces from chunks of OCR output [16], while SV-RAG [12] and M3DocRAG [13] retrieve relevant page images.

We find these existing methods all fall short in understanding semi-structured documents, like illustrated in the example below.

Example 1.2. Continue with Example 1.1, when given a document and a question like the one shown in Figure 1, we observe that ZenDB fails to answer primarily because it breaks the structure of Table 1 by flattening it into plain text under Chapter 3. This disruption makes it difficult to align the text paragraph mentioning "winter" with the relevant table region (Decoquinate, Experiment 3) needed to identify the correct answer (3.20). In addition, ZenDB misidentifies the document hierarchy (e.g., nesting Chapter 3 under Chapter 2) and retrieves overly redundant content (e.g., the entire document), further impeding it to give the correct answer. Similarly, VLM (GPT-5) hallucinates an answer (3.8) by only analyzing the table image, while overlooking the surrounding paragraph that indicates Experiment 3 was conducted in winter. RAG (GPT-5) retrieves the sentence describing how the average feather score

was computed, but fails to align the retrieved Table 1 (in HTML-rendered chunk) and produces an incorrect value (3.26). In contrast, we aim to aggregate elements into semantically meaningful units (e.g., tables with titles and positions) and explicitly capture their relationships and layout distinctions, so as to enable joint reasoning (e.g., the target paragraph and table row) and give correct answer.

Challenges. The technical challenges of this work are threefold.

First, the elements extracted by OCR (e.g., paragraphs, tables, charts) are often fragmented and detached from their semantic context, making it difficult to interpret them in isolation. This hinders fine-grained document structure understanding (i.e., component level), such as associating a chart with its title or aligning a paragraph with a nearby table (**C1**).

Second, existing approaches lack effective representations to capture hierarchical structures (e.g., tables under nested chapter titles) and to maintain layout-specific distinctions (e.g., differentiating sidebars from main content). For instance, while scientific reports follow a nested chapter structure with interleaved visual elements, newspapers often contain parallel articles. Current methods either treat documents as flat sequences [16, 26] or process them page-wise [12, 13], thus failing to model global structure (document level) and cross-component relationships (**C2**).

Third, answering complex questions requires retrieving and aligning evidence scattered across multiple regions or pages. For instance, in Figure 1, the answer requires (1) identifying an experiment ID from the question, (2) linking it to a season mentioned in a paragraph, and (3) locating the corresponding value in table within another page. Moreover, retrieval must support both semantic references (e.g., summary of nested chapters) and spatial ones like *"top of page 1"* (**C3**).

Our Methodology. To address these challenges, we propose `MoDora`, an LLM-powered system for semi-structured document analysis. First, we propose a local-alignment aggregation strategy that transforms OCR-parsed elements into self-contained components, and enriches these components by (1) detecting hierarchical structures of paragraph titles and (2) extracting semantic triples ($title$, $metadata$, $data$) for non-text elements (e.g., tables, charts). Second, we design the *Component-Correlation Tree* (CCTree) to hierarchically organize document components, which can explicitly model inter-component relations and layout-aware distinctions (e.g., separating footnotes and sidebars). We construct CCTree using a bottom-up cascade summarization strategy to propagate concise content upward through the tree. Finally, we employ a question-type-aware retrieval mechanism that supports (1) LLM-guided node selection by titles and metadata, (2) embedding-based fallback to recover missed evidence, and (3) an LLM-based verifier to validate the selected nodes through cross-modal reasoning.

Contributions. This paper makes the following contributions:

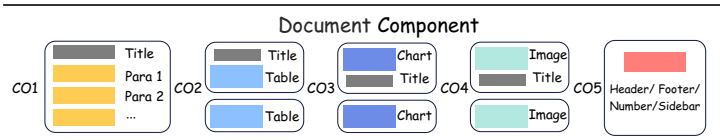
(1) We propose `MoDora`, a novel semi-structured document analysis system that supports diverse document layouts and handles both semantic and location-based questions (see Section 3).

(2) We design a local-alignment aggregation strategy that transforms OCR-parsed elements into self-contained components, incorporating MLLM-based hierarchy detection for titles and structured semantic extraction for non-text elements. (see Section 4).

(3) We introduce the Component-Correlation Tree (CCTree), a hierarchical representation that captures inter-component relationships and layout-aware distinctions, and enables bottom-up cascade summarization to enhance contextual reasoning (see Section 5).

(4) We propose a tree-based data retrieval strategy that integrates question-type-aware retrieval strategy, LLM-guided node selection, embedding-based fallback, and MLLM-based cross-modal verification to support robust evidence localization (see Section 6).

(5) Our extensive experiments on standard and specific benchmarks demonstrate significant accuracy improvement (ranging 5.97% to 61.07%) over existing methods (see Section 8).



Document Type	Reading Order (in sequence)	Page-Level Layout	Example Component Correlation
research paper	{ CO5, CO5, CO1, CO2, CO1, CO1, CO1, CO3, CO5 }	single column	4 hierarchical titles and their paragraphs (CO1s)
magazine	{ CO1, CO3, CO1, CO4, CO1, CO4, CO5 }	mixed double- and single-columns	1 Chart (CO4) nested in 1 paragraph (CO1)
poster	{ CO1, CO3, CO4, CO1, CO2, CO2, CO1, CO2, CO1 }	multiple columns	1 top-level title with 1 Chart (CO4) and 3 sub-titles (CO1s)

Fig. 2. Example Semi-structured Document Layouts.

2 Preliminaries

In this section, we introduce the concepts and definition of semi-structured documents (Section 2.1), formalize the definition of semi-structured document analysis (Section 2.2), and discuss the limitations of existing work using a pilot study (Section 2.3).

2.1 Semi-Structured Document

Document Elements. In this work, we define a *document element*, written as $\boldsymbol{e}$, as the atomic unit of a document $\mathcal{D}$, belonging to one of the following five primary categories:

- ***Text*** ($\boldsymbol{e^t}$) includes both (i) paragraphs that constitute the main body of the document (e.g., the two paragraphs about experimental design in Figure 1) and (ii) titles at different hierarchical levels, which represent the theme or summarize the content. For example, in Figure 1, the title "Experiment ..." precedes several paragraphs across page 1 and 2, while the title "Weekly trend of feather score in Exp.1" leads a chart.
- ***Tables*** ($\boldsymbol{e^b}$), including structured tables and semi-structured tables [23], store attributes and values in cells and reflect relational or nested relations through the spatial arrangement of cells. Tables in semi-structured documents are relatively small in scale but usually contain key records or statistical information for analysis.
- ***Charts*** ($\boldsymbol{e^c}$), such as bar charts and line charts, are visual representations of data tables, expressing data relationships through graphical forms (e.g., bars, curves, and sectors), labels, legends, and axes (e.g., scales and grids).
- ***Images*** ($\boldsymbol{e^i}$) encompass diverse visual content such as photos, maps, and flow diagrams, representing specific scenes or events.
- ***Supplements*** ($\boldsymbol{e^s}$) include headers, footers, sidebars, and page numbers outside main body, providing auxiliary metadata or navigational cues (e.g., headers with chapter titles, external links).

Document Components. We define a document component (CO) as the minimal, self-contained unit of a document, consisting of one or more semantically and visually coherent elements, written as $CO = \{e_k^x\},\ CO \subseteq \mathcal{D}$, where e_k^x denotes an atomic document element. For example, as shown in Figure 2, a title followed by multiple paragraphs is grouped as CO1, while a table and its title are aggregated into a table-related component (CO2).

Semi-Structured Document. A semi-structured document consists of a sequence of layout-ordered components, i.e., $\mathcal{D} = \{CO_1, CO_2, \ldots, CO_n\}$, where each CO_i is formed by one or more tightly coupled elements, collectively capturing the document's text and structure information.

As shown in Figure 2, a semi-structured document exhibits all of the following key characteristics: (1) *Diverse element types:* Such documents often contain multiple types of elements within the same page, distinguishing them from pure unstructured text data. (2) *Complex cross-element relationships:* There exist structural and semantic associations among elements, such as hierarchical title structures, spatial grouping, and cross-references between text and tables. (3) *Long document ranges:* The document typically spans a relatively long range (often dozens of pages). Unlike long plain text [10], we focus on challenges in modeling and inference arising from multi-modal elements and complex layouts.

2.2 Semi-Structured Document Analysis

Given a semi-structured document $\mathcal{D}$ composed of heterogeneous components (e.g., text, tables, charts) and a natural language question Q, the goal is to retrieve a subset of components from $\mathcal{D}$ that are relevant to Q, such that models can reason over the retrieved components to produce a final answer $\mathcal{A}$.

Definition 1 (Semi-Structured Document Analysis). *Formally, this process is defined as a function* $f : (\mathcal{D}, Q) \mapsto \mathcal{A}$, *where the answer* $\mathcal{A}$ *is derived based on a set of components* $C_{rel} = \{CO_{k_1}, CO_{k_2}, \ldots, CO_{k_m} \mid CO_{k_i} \subseteq \mathcal{D}, CO_{k_i} \text{is relevant to } Q\}$.

Drawing on existing benchmarks and real-world data (see Figure 7), we identify four representative question types in this task.

(1) Text-Only Questions only need the semantics of unstructured text in the document to answer correctly. For example, to answer the question "*What temperature environment is experiment 3 conducted in?*", we first retrieve the text component with the title "*3. Experiment Design and Results*" , and then perform semantic reasoning over its content to derive the answer "*0-15 degrees*".

(2) Hybrid Data Questions need semantic information beyond unstructured text in the document, such as that contained in tables, charts, and images. For example, to answer "*What is the feather score of Decoquinate group in Exp.3?*", we first identify the component containing "*TABLE 1...*" by semantically interpreting its title and content, and then perform reasoning over the structured data in the table to answer "*3.20*".

(3) Structural Hierarchy Questions require understanding the hierarchical structures of the document to answer correctly. For example, to answer "*From which two aspects does the article analyze the possible causes of the monensin effects?*", the system should identify "*Feather Lesions*"and "*Changes in Nutritional Requirements*" as the two parallel chapter titles under the "*Result Analysis*", distinguishing them from other textual content. Such questions aim to extract key information or summaries from multiple document chapters [34], rather than to retrieving detailed paragraph content.

(4) Location-Based Questions retrieve components based on specific locations rather than semantics. For example, to answer "*What is at the bottom-right of page 1?*", we filter components that meet the criteria based on their location information, including page indices and bounding box

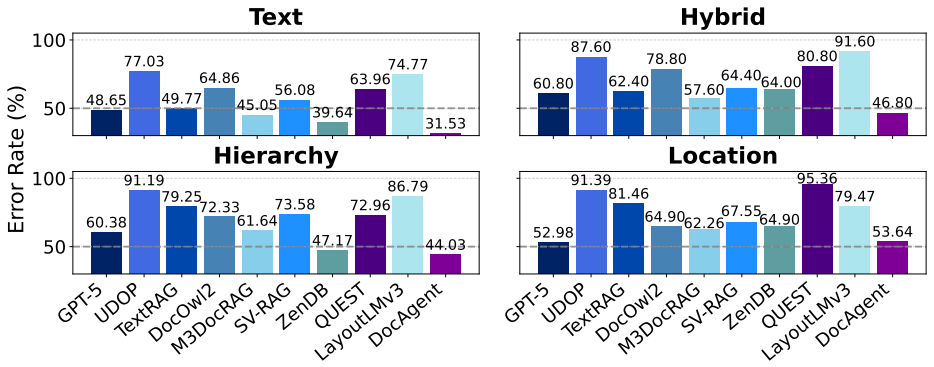


Fig. 3. Error Rate Distribution of Existing Methods.

coordinates. The result in this case is the figure title "*Figure 1...*" and the figure containing two bar plots.

We would like to note that (1) `MoDora` centers on structurally modeling document elements and question answering over them. For document parsing that extracts elements from the document, we adopt existing OCR techniques (see Section 4), which is not our focus. (2) `MoDora` aims to effectively handle the four representative question types introduced above (see Section 8.2), while also supporting more complex multi-hop ones (e.g., involving value aggregations and filtering). Such questions can often be decomposed into combinations of the basic types, for which existing query decomposition techniques [27] can be readily applied.

2.3 Limitations of Existing Methods

Despite recent progress in document analysis, our pilot study (Figure 3) reveals that existing methods remain inadequate for semi-structured documents, particularly in *capturing visual, hierarchical, and layout-specific structures*, which we summarize below.

Content Extraction Methods. As discussed in Section 1, methods such as QUEST [26] and EVAPORATE [9] transform documents into relational tables to support structured querying. We find QUEST performs poorly on hierarchical and spatial tasks. The main reason is its failure to preserve structural context during extraction. Moreover, the gap between natural language questions and SQL-style relational queries hinders answer generation.

Structure Extraction Methods. ZenDB [20] and DocAgent [25] achieve relatively good accuracy. ZenDB uses visual feature clustering and LLMs to infer partial structures (e.g., separating titles and body text, detecting title levels). However, its error rate of hierarchy analysis approaches 50%, due to the unreliability of visual feature clustering and weak generalization to non-digital documents. Moreover, its focus on text and titles makes it prone to errors in hybrid data analysis involving tables and charts. In contrast, we aim to natively support multi-modal documents and can uniformly process those without native PDF structures. DocAgent extracts document content into an XML-based tree outline, which is used to guide the agent's actions and reviews via predefined tools (e.g., keyword search, page and chapter IDs). However, the execution of actions and reviews is only loosely coupled with the XML tree, and the node relationships are largely underutilized. As a result, DocAgent performs poorly on tasks that require structural and layout-aware reasoning, such as hierarchy recognition and location-based analysis, compared to its performance on purely textual tasks.

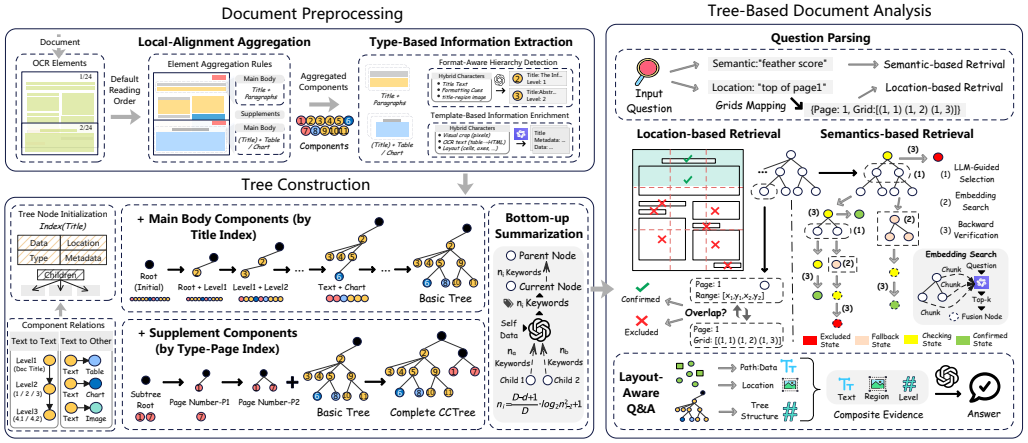


Fig. 4. System Overview of MoDora.

End-to-End Models. End-to-end models, including LayoutLMv3 [18], DocFormer [8], UDOP [28], DocOwl2 [17], and GPT-5 [22]), enable end-to-end reasoning through pre-trained models or MLLMs. LayoutLMv3, DocFormer and UDOP natively support page-level processing with hybrid input of images, OCR-parsed text, and bounding boxes. They employ techniques like discrete multi-modal encoder [8], word-patch alignment [18] and vision-text-layout unified encoder [28]. However, limitations in model scale, maximum sequence length (e.g. 512 for text tokenization in LayoutLMv3) and insufficient high-level document structure modeling restrict their performance, particularly on semi-structured documents. DocOwl2 [17] addresses some of these limitations by incorporating high-resolution visual compression and utilizing a more robust LLM backbone, enabling multi-page input and improved performance, but it still lags behind more advanced MLLMs like GPT-5. Nonetheless, GPT-5 also faces notable challenges in processing semi-structured documents. It often overlooks critical details and struggles to reason over cross-element relationships, such as aligning tables with corresponding paragraphs. Consequently, its performance falls short of structure-aware methods like DocAgent.

RAG-Based Methods. Methods such as TextRAG [16], SV-RAG [12], and M3DocRAG [13] retrieve coarse-grained document segments (e.g., pages or chunks) to mitigate input length limits. We find TextRAG underperforms GPT-5 as it suffers from semantic loss during OCR-based text extraction. M3DocRAG improves performance on text and hybrid tasks by incorporating refined retrieval, but fails to capture hierarchical and locational cues, where its visual encoder lacks the structural grounding needed for these tasks.

3 System Overview

As shown in Figure 4, MoDora is composed of three main modules.

Document Preprocessing. To transform raw OCR outputs into self-contained components, we first apply a local-alignment aggregation strategy that groups fragmented elements into layout-aware components. We then extract enriched semantics for two key component types: (1) *Hierarchical titles*, whose hierarchical levels are inferred using a format-aware LLM prompt that incorporates both semantic and visual patterns; and (2) *Non-text elements* (e.g., tables, charts), for which we generate structured semantic triples $(title, metadata, data)$ for information enrichment.

Tree Construction. To capture document-level structure beyond linear reading order, we propose the Component Correlation Tree (CCTree), where each document component is represented as a node. We construct CCTree in two stages: (1) *Main content organization*, which groups components based on detected title hierarchies and semantic links between textual and nearby non-text elements (e.g., tables, charts); and (2) *Supplementary content isolation*, which attaches auxiliary components (e.g., sidebars) as separate subtrees to prevent semantic interference. For efficient retrieval, we adopt a bottom-up summarization process that recursively propagates subtree summaries upward, with their length constrained by an empirical formula to balance informativeness and redundancy.

Tree-Based Document Analysis. We design a *question-type-aware retrieval strategy* over the CCTree to accurately locate relevant components. We first parse locational and semantic cues in the question and adopt different retrieval strategies accordingly: (1) For *location-based questions*, we select nodes based on spatial metadata, aligning with positional cues in the question. (2) For *semantics-based questions*, we perform a forward search for pruning strategy on CCTree through LLM selection based on node index and metadata, and further apply embedding search as the fallback to recover potential relevant information in unselected subtrees. The MLLM backward verification based on detailed contents and locations is then used to eliminate false positives in LLM selection and embedding search. The final set of evidence along with the CCTree's structure is passed to an MLLM to generate the answer.

4 Document Preprocessing

Document preprocessing aims to transform raw document content into structured and semantically meaningful elements, bridging the gap between low-level visual components (e.g., text boxes, tables, charts) and high-level document structures (e.g., chapters, paragraphs, data-bearing components). In this work, we assume as input a sequence of OCR-parsed elements extracted by tools like PaddleOCR [14], enabling uniform processing across both digital-native and rasterized documents. Compared to native PDF parsers [5, 7], OCR offers broader applicability and retains key typographic cues (e.g., font size, indentation) that are essential for document structure modeling. Based on these OCR-parsed elements, we systematically address three main challenges.

(1) *Integrating OCR-detected elements into coherent components is nontrivial.* For example, in Figure 1, the title "Abstract" and its subsequent paragraphs should be merged into a single component, whereas the title "Fig. 1: Comparison $\cdots$" should be combined with its corresponding bar charts. (2) *Establishing the hierarchical structure of components is essential.* For instance, in Figure 1, the components "Abstract" and "Introduction" should be placed at the same hierarchical level under the article title. (3) *Enriching the attributes of non-text components remains challenging.* Tables, charts and images may lack associated textual metadata such as titles or descriptions, and relevant information must be extracted from the visual or structural content of the elements.

To address these challenges, we propose three key techniques: (1) a local-alignment aggregation strategy to detect and organize document components, (2) a format-aware hierarchical relationship detection method to model structural relationships among components, and (3) a template-based information enrichment approach to enrich components containing non-textual elements.

4.1 Local-Alignment Element Aggregation

Direct document analysis at the element level faces challenges in retrieval and reasoning, primarily due to its inability to capture inter-relationships among elements. To address this limitation, we introduce the concept of component ($CO = \{e_k^x\}, CO \subseteq \mathcal{D}$) in section 2.1. as a group of locally interrelated elements, to facilitate more coherent understanding of document structures. However, the diverse characteristics of elements in semi-structured documents make it challenging to integrate them into components. For instance, adjacent paragraphs under a common title can be concatenated

in reading order, but when they span multiple pages, intervening headers or footers may disrupt semantic continuity and cause ambiguity. Moreover, tables, charts and images that appear between paragraphs differ from independent headers and footers, as they are typically titled and linked to the surrounding text.

To tackle the mentioned challenges, documents are first processed by OCR models (e.g., PaddleOCR [14]) to produce sequences of elements arranged in human-preferred reading order. Since the reading order captures the sequential relationships among elements and the element types are identified, it is intuitive to segment the sequence into intervals and organize each subsequence into a specific component. To differentiate various local element layouts, we apply several heuristic rules, as illustrated in Figure 4. Here we introduce several frequently used rules.

(1) Textual Aggregation: Typically, multiple adjacent paragraphs headed by a title exhibit semantic coherence and thus form a natural component. We aggregate a title and all paragraphs that appear between it and the subsequent title into a component, which can be represented as $A_1 = \{e^t_{title}, e^t_{para,1}, \ldots, e^t_{para,n}\}$. Each component uses the title as its identifier, concatenates all associated paragraphs as its content, and records the spatial information of its constituent elements as its location, including page indices and bounding box coordinates. For example, in Figure 1, the title "Abstract" and the paragraphs between it and "Introduction" constitute a component.

(2) Table/Chart Aggregation: For tables, charts and images, their titles (different from those of documents or paragraphs) may appear either before or after them, or in some cases, be absent entirely. For example, if the element immediately preceding a table is identified as a table title and has not yet been assigned to another component, it is designated as the table's title. Otherwise, if the subsequent element is recognized as a table title, the table adopts it as its title. In the absence of a surrounding title for tables, charts and images, a default title is assigned to them. The aggregated component can be represented by $A_2 = \{[e^t_{title}], e^x\} \cup \{e^x, [e^t_{title}]\}, x \in \{b, c, i\}$, where $[e^t_{title}]$ represents that the title is optional according to different situations mentioned above. The locations of constituent elements are also recorded as part of the component's attributes. For example, in Figure 1, the title "Figure 1: Comparison..." and the bar plots above it form a component while the title "Table 1: Effect..." and the table below it form another component.

(3) Supplement Element Aggregation: Documents also contain various supplementary elements such as headers, footers, sidebars, and page numbers, which are typically independent on each page. Therefore, each type of supplementary elements on each page is converted into a component and can be represented by $A_3 = \{e^s\}$. The location information of these elements is also recorded within their corresponding component attributes. For example, the page number "1/24" and "2/24" in Figure 1 each form a component.

4.2 Format-Aware Hierarchy Relation Detection

Text elements inherently exhibit a nested structure through hierarchical titles, corresponding to the hierarchical organization of the CCTree. However, this structural information is often lost in the linear sequence of elements, and the diverse characteristics of document elements further complicate hierarchy detection. To address this challenge, we propose a format-aware approach for identifying hierarchical relationships among elements.

With the components obtained through local-alignment aggregation, the task of hierarchy detection is mainly about determining the nested levels of paragraphs. Since each component obtained by rule A_1 has a title, identifying the level of the title effectively determines that of the entire component.

Intuitively, title levels are determined based on both semantic and textual formatting information. To this end, we extract the list of titles' text and their corresponding locations from the document components to form $\{e^t_{title,1}, \ldots, e^t_{title,n}\}$. The recorded locations are used to crop the document

regions of each title, which are then concatenated into a unified title sequence vertically. Next we input both the text and image title sequence to an MLLM (e.g., GPT-5) and instruct it to identify the hierarchical level of each title with the input (i.e., represented by the number of "#" symbols).

4.3 Template-Based Information Enrichment

As mentioned above, tables, charts and images can lack explicit titles and are therefore assigned default ones. Moreover, certain information is conveyed in non-textual forms, such as table structures and charts. Due to the diverse structures and rich semantics of these elements, comprehensively extracting their integrated information is challenging. To address this, we propose a template-based method to guide the extraction of such information.

Specifically, we define three types of attributes to facilitate the progressive extraction of integrated information, forming a triplet $\mathcal{T}(CO) = \big(title(CO), metadata(CO), data(CO)\big), CO \subset A_2$. We prompt MLLM (e.g., Qwen2.5-VL) to generate the triplet in one call. Specifically, the prompt provides differentiated few-shot templates for different primary element. For example, for charts, the prompt instructs the model to focus on axes, labels, and visual trends; whereas for tables, the prompt emphasizes the extraction of schema and representative cell values. Following this, each component possesses the title, data, and location attributes, along with component-specific attributes reflecting its unique composition.

5 Tree-Based Document Modeling

5.1 Component Correlation Tree

The resulting component sequence does not fully preserve the original document layout (e.g., nested hierarchies or the boundaries between main content and supplementary elements) and thus requires a more structured representation that captures high-level correlations. To address this, we propose the Component Correlation Tree (CCTree), which models the relationships between document components while enabling efficient content retrieval.

Nodes of CCTree. As the fundamental unit of the tree, each node $v \in \mathcal{V}$ in the CCTree represents a document component, retaining the component information and extended with two attributes to facilitate efficient retrieval: (i) node index v_{index} as a unique identifier and (ii) children v_{child} recording all child nodes.

Edges of CCTree. Edges $\epsilon \in \mathcal{E}$ in the CCTree represent the relationships between nodes in three types.

(1) ***Text-to-Text Relationship.*** As the title levels of text paragraphs define the nested hierarchy of a document, the results of format-aware hierarchy detection can be used to establish these relationships. A text component with a higher-level title serves as the parent, whereas components with lower-level titles are designated as children. The relationship can be represented as $\mathcal{E}_1 = \{(CO_i, CO_j) \mid CO_i, CO_j \subset A_1\}$.

(2) ***Text-to-Other Relationship.*** Since non-text components (e.g., tables and charts) typically share a similar semantic theme with adjacent text-based components, we regard them as complementary to the surrounding text. Accordingly, we model the text-based component as the parent node, while the adjacent non-text components are designated as its children. The relationship can be represented as $\mathcal{E}_2 = \{(CO_i, CO_j) \mid CO_i \subset A_1, CO_j \subset A_2\}$.

(3) ***Independent Supplementary Relationship.*** For components containing supplementary elements (e.g., page headers and footers) that are not directly related to the main content, we assign them to an independent branch of the CCTree, using the virtual root node as their parent. The relationship can be represented as $\mathcal{E}_3 = \{(CO_i, CO_j) \mid CO_i, CO_j \subset A_3\}$.

5.2 Two-Stage Tree Construction

With the defined relationships between tree nodes, we construct the CCTree through a two-stage connection establishment. In the first stage, text-to-text and text-to-other relationships are established, followed by the second stage, which handles the independent supplementary relationships. Algorithm 1 shows the overall process.

Algorithm 1: CCTree Construction.

Input: An ordered list L of components
Output: The CCTree rooted at node v_r

$L_1 \leftarrow [\,]$;
$L_2 \leftarrow [\,]$;
for *each* CO *in* L **do**
  if e^s *in* $CO.elements$ **then**
    $L_2.append(CO)$;
  end
  else
    $L_1.append(CO)$;
  end
end
$v_r \leftarrow \mathcal{V}.init()$;
$v_r.add\_child(Main\_Branch(L_1))$;
$v_r.add\_child(Supp\_Branch(L_2))$;
return v_r ;

Tree Construction with Main Body Components. The construction for main body components is based on three key factors: the components reading order, their types, and the title levels, as illustrated in Algorithm 2. It begins by creating a root node with the highest hierarchical level and pushing it onto a stack. Components are then processed sequentially according to the reading order. For a text component, if its title level is lower than that of the node on the stack top, it is linked as the child of the top node and subsequently pushed onto the stack. Otherwise, nodes are continuously popped from the stack until the top node has a higher level than the current component, after which the current component is linked and pushed. For a non-text component, it is directly linked as a child of the top node without being pushed.

Figure 4 shows an example. The basic tree of main content is constructed by title index, where the second level title "*Abstract*" is linked as the child of first level title "*The Inf...*", and the table "*Table 1...*" is linked as the child of "*Experiment ...*".

Tree Construction with Supplement Components. The first stage does not process components of supplementary elements, as they are relatively independent and unsuitable for reading-order-based construction. In the second stage, these components are linked to a separate branch according to their type and page attributes. The detailed procedure is presented in Algorithm 3.

For the example document in Figure 1, the supplementary branch with page number "*1/24*" and "*2/24*" is gradually constructed by type-page index. This branch is not correlated with any components from the main content as shown in Figure 4.

Algorithm 2: Main Body Branch Construction $Main\_Branch()$.

Input: An ordered list L of components
Output: The root node r of the basic tree

$v_r \leftarrow \mathcal{V}.init(title\_level = 0)$;
$S \leftarrow stack.init()$;
$S.push(v_r)$;
for *each* CO *in* L **do**
  if $CO.elements \cap \{e^b, e^c, e^i\} = \varnothing$ **then**
    $v \leftarrow \mathcal{V}.init(CO)$;
    while $S.top().title\_level >= v.title\_level$ **do**
      $S.pop()$;
    end
    $p \leftarrow S.top()$;
    $p.add\_child(v)$;
    $S.push(v)$;
  end
  else
    $p \leftarrow S.top()$;
    $p.add\_child(v)$;
  end
end
return r ;

Algorithm 3: Supplementary Branch Construction $Supp\_Branch()$.

Input: An ordered list L of components, basic tree root r
Output: The root node v_r of the document tree

$v_r \leftarrow \mathcal{V}.init(children = [\text{“}header\text{”}, \text{“}footer\text{”}, \text{“}sidebar\text{”}, \text{“}number\text{”}])$;
for *each* CO *in* L **do**
  $v \leftarrow \mathcal{V}.init(CO)$;
  $v_r.children[v.element\_type].add\_child(v)$;
end
return v_r ;

5.3 Bottom-up Metadata Summarization.

The document tree construction process connects all components into a complete CCTree. However, the resulting structure is not yet suitable for tree-based retrieval, as higher-level nodes do not inherit information from their lower-level children. Therefore, we propose a bottom-up metadata summarization method that integrates lower-level information into higher-level nodes.

The process starts at the leaf nodes of the CCTree, where each node invokes an LLM to generate several nominal phrases serving as metadata that comprehensively summarize its content. For non-leaf nodes, metadata generation additionally incorporates the metadata of all child nodes. Within the same hierarchical level, each node corresponds to a subtree that contains varying amounts of information. During upward information propagation, it is difficult to ensure that all information is completely preserved, leading to inevitable information attenuation. To quantitatively measure and

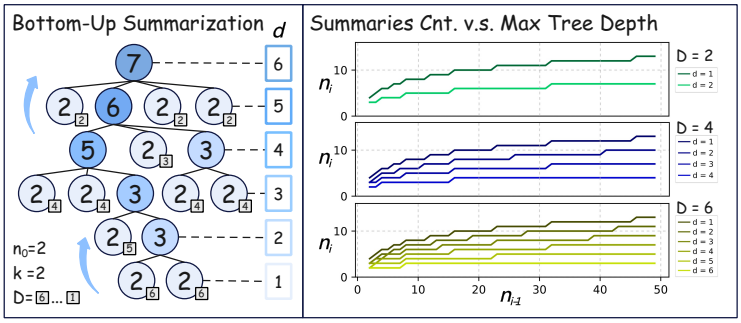


Fig. 5. The number distribution of summary keywords in tree nodes with different depth and subtree sizes.

assign the amount of summarized information, we define the summarization as several keywords, and the number of which for each node follows the information attenuation formula:

$$n_i = \frac{D_i + d_i - 1}{D_i} log_2(n_{i-1}^k) + 1$$

where D_i denotes the maximum depth of the tree that includes node v_i, d_i represents the depth of node v_i, n indicates the number of summaries of node v_i, and n_{i-1} denotes the total number of summaries generated by the children of node v_i. The term $\frac{D_i+d_i-1}{D_i}$ serves as the information attenuation coefficient, while $log_2(n_{i-1}^k)$ represents the amount of transmitting information which has a natural decay rate, with k controlling the rate of information growth.

The information attenuation formula determines the number of summaries allocated to each node. After the recursive summarization process, each node retains metadata that encapsulates the semantic content of its entire subtree. The left side of Figure 5 illustrates the bottom-up summarization process with $k = 2$, where the number within each node denotes its assigned summaries. The right side compares the number of summaries under different maximum tree depths, revealing the information attenuation effect. For instance, when a parent node aggregates 12 child summaries, it generates 7 summaries after attenuation.

It is worth noting that metadata for non-textual elements have already been generated during the template-based integrated information completion process. To avoid redundant LLM calls and improve efficiency, the existing metadata of these nodes are directly utilized when they serve as leaf nodes.

6 Tree-Based Document Analysis

In the document tree construction, the entire document is represented as a CCTree, where each node stores textual and locational information. Document analysis involves retrieving relevant evidence nodes and reasoning upon them. The main challenge lies in leveraging the tree structure to optimize retrieval, chasing higher recall and precision. Several retrieval strategies can be applied, each with inherent limitations. Node-by-node traversal guarantees high recall and precision but suffers from high computational cost. Pruning strategies improve efficiency by excluding irrelevant branches but risk missing evidence in pruned subtrees. Embedding-based retrieval offers higher efficiency by directly capturing fine-grained semantics, yet it depends on appropriate chunking and fails to capture long-distance context dependencies.

Considering the strengths and limitations of the above methods, we propose a CCTree retrieval algorithm that integrates MLLM reasoning, pruning strategies, and embedding-based search to balance recall and precision.

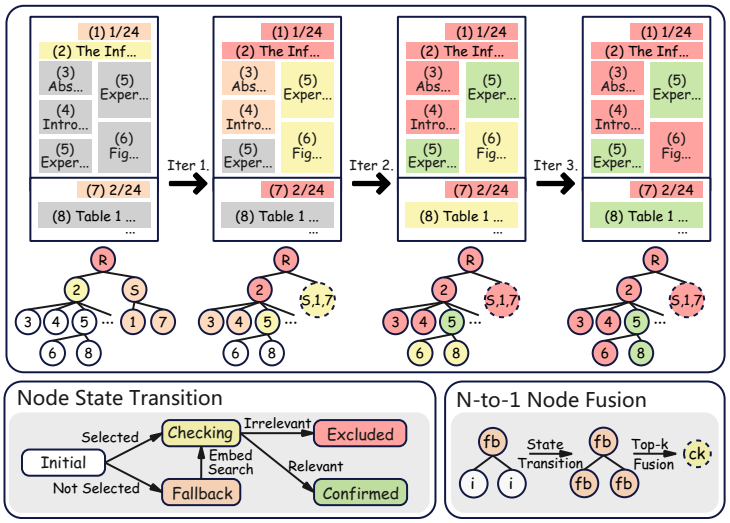


Fig. 6. Tree-based Data Retrieval.

Semantics- and Location-Aware Retrieval. Locational questions requires retrieval based on spatial location rather than semantics. In the CCTree, locations of document elements are stored in the location attributes of nodes, while semantic information is in other attributes. To handle different question types, we adapts location-based retrieval and semantics-based retrieval for locational and semantic cues in the question respectively. Given a document tree, the retrieval algorithm is to evidence nodes in the form of `(path-data)` pairs and `(path-location)` pairs. Here, the path represents the sequence of indices from the root to the target node in CCTree, while the data and location are corresponding attributes stored in that node.

(1) Location-Based Retrieval. During location-based retrieval, `MoDora` extracts components whose locations intersect with regions specified in the original question. Specifically, each document page is divided into 3×3 grids, and each grid is represented as a tuple $(row, column)$ (e.g., `(1,1)` corresponds to the top-left, `(3,2)` to the middle-bottom). Locational cues from the raw question are then mapped to one or more of these grids (e.g., "bottom of the page" maps to `[(3,1),(3,2),(3,3)]`). The CCTree is subsequently traversed, and any node whose location overlaps with the mapped regions is designated as one of the evidence nodes.

(2) Semantics-Based Retrieval. During semantics-based retrieval, `MoDora` performs two iterative steps: (i) forward search executes pruning by deciding whether a node should be added to the candidate evidence node list, which combines a loose LLM selection mechanism with a fallback of embedding search to achieve high retrieval recall, and (ii) backward verification decides whether a node in the candidate list is confirmed as an evidence node or discarded, which leverages MLLM as the verifier to enhance the precision. Figure 6 shows the retrieval process for the document and question in Figure 1, where different colors represent different node states.

Semantics-based retrieval begins by adding the CCTree root to the candidate evidence node list and initiates an iterative process. In each iteration, forward search and backward verification are performed on each node in the candidate list. The algorithm terminates and returns the results once the candidate list is empty. As shown in the bottom of Figure 6, throughout the retrieval process, each node can occupy one of five possible states, including (i) Initial State ($\mathcal{S}^i$), where the node has not yet been visited, (ii) Checking State ($\mathcal{S}^{ck}$), where the node has been selected by the forward search or returned by embedding search, and is awaiting backward verification, (iii) Fallback State ($\mathcal{S}^{fb}$), where the node was not selected by forward search and is scheduled for

processing via embedding search. (*iv*) Confirmed State ($\mathcal{S}^{cf}$), where the node is determined as relevant by the backward verification as an evidence node. (*v*) Excluded State ($\mathcal{S}^{ex}$), where the node is determined as irrelevant by the backward verification to be a non-evidence node.

Step 1. Forward Search. For nodes in the candidate evidence list, forward search determines whether each of its child nodes should be added to the list. We prompt the LLM with the question, along with the index and metadata of each child node, to generate a list of indices corresponding to the selected nodes. The selected nodes are those that the LLM deems relevant to the question based on the index and metadata. They are then added to the candidate evidence list, and their state is updated from initial to checking ($\mathcal{S}^{i} \rightarrow \mathcal{S}^{ck}$).

However, since node indices and metadata cannot fully capture the semantics of each node, we employ a fallback of embedding search to capture potential evidence from the sub-trees rooted at nodes not selected during the LLM selection, thereby enhancing recall. As shown in the bottom-right of Figure 6, for each node not selected by the LLM, its state and those of all its descendant nodes are updated from initial to fallback ($\mathcal{S}^{i} \rightarrow \mathcal{S}^{fb}$), and the corresponding sub-tree is then flattened, chunked, and embedded for retrieval. The *top-k* content pieces most relevant to the question are merged into a fusion node without children, which is subsequently added to the candidate evidence node list ($\mathcal{S}^{fb} \rightarrow \mathcal{S}^{ck}$).

Step 2. Backward Verification. The forward search aims to maximize recall by adding as many potential evidence nodes as possible to the candidate list. However, the question may only share certain associations with a node's index or metadata, or the node may be incorrectly retrieved during the embedding search. To eliminate false positives and ensure retrieval precision, it is therefore necessary to examine the detailed content of each node, which we refer to backward verification.

Specifically, we perform backward verification on each node within the candidate list. An MLLM is prompted to determine whether the node data contains evidence relevant to the question. If the judgment is positive, the node's state is updated to confirmed ($\mathcal{S}^{ck} \rightarrow \mathcal{S}^{cf}$) and we output the node. Otherwise, it is just removed from the candidate list and its state is set to excluded ($\mathcal{S}^{ck} \rightarrow \mathcal{S}^{ex}$). For an actual CCTree node, both its textual content and the corresponding document region cropped according to its location are provided as inputs to the MLLM. For a fusion node, only its textual content retrieved through the embedding search is used.

Evidence Aggregation and Answer Generation. Finally, we aggregate the results of semantics- and location-aware retrieval into a unified representation to perform final answer generation. Specifically, three types of evidence are aggregated from the retrieved CCTree nodes. If the retrieval results are empty or deemed irrelevant to the question by the answering model, all nodes in the CCTree will be used to ensure completeness.

(1) ***Textual Evidence.*** To enable the MLLM to achieve a fine-grained understanding of the retrieved results, the retrieved CCTree nodes are first converted into (path-data) pairs, forming a list of textual evidence to be provided to the MLLM as part of its input.

(2) ***Locational Evidence.*** Although textual evidence is also extracted for non-textual elements, it fails to capture the structural and formatting information embedded within these elements (e.g., table and chart layouts, font color and size). So we introduce locational evidence to enhance answer generation. Specifically, we crop corresponding document regions with the locations stored in retrieved tree nodes, which is incorporated into the MLLM input.

(3) ***Hierarchical Information.*** Both textual and locational evidence provide question-relevant information. However, the global context of the document also plays a crucial role. Therefore, we incorporate the index schema of the CCTree, which represent the document's hierarchical structure, into the MLLM input as global contextual information.

Using the original question together with the aggregated textual evidence, locational evidence, and hierarchical information, we prompt the MLLM to generate the final answer.

7 Micro Benchmark Preparation

Existing document datasets do not sufficiently cover typical semi-structured document analysis tasks [13, 30, 31]. The documents in MP-DocVQA [30] and DUDE [31] exhibit limited content and structural complexity. While M3DocVQA [13] poses greater structural challenges, its Wikipedia-sourced documents lack element diversity and fail to reflect broader real-world document types. To address these gaps, we introduce a new benchmark MMDA.

Document Diversity Metrics. Since basic metrics (e.g., number of pages and elements) fail to capture a document's layout complexity and element diversity, we introduce two new metrics to characterize structural richness: Element Diversity (*ED*), which measures the diversity of element types, and Layout Diversity (*LD*), which quantifies the variance in the components' locational distribution.

(1) Element Diversity (ED). A document dominated by any single type (e.g., tables, text) tends to exhibit low diversity. Thus we define Element Diversity (*ED*) as the normalized standard variance in the counts of the five element categories (see Section 2), i.e.,

$$ED = \sqrt{\frac{1}{n}\sum_{x}(e^x - \frac{1}{n}\sum_{y} e^y)^2} \Bigg/ \sum_{x} e^x, \quad where \;\; x, y \in \{t, b, c, i, s\},$$

where t, b, c, i, s denote text, table, chart, image, and supplement respectively and n is the number of total elements. A larger *ED* value indicates lower element diversity.

(2) Layout Diversity (LD). While *ED* measures content variety, it does not reflect structural composition. We therefore define *Layout Diversity (LD)* to quantify the average number of independent components per page, i.e., $LD = \frac{N_{\text{comp}}}{N_{\text{page}}}$, where N_{comp} denotes the number of independent components and N_{page} the total number of pages. An independent component refers to a contiguous sequence of elements of the same type; for example, four adjacent tables are counted as one independent component. A higher *LD* indicates richer layout organization and stronger structural variation.

Dataset Preparation. To construct the MMDA benchmark, we begin by sampling 200k candidate documents from over one million real-world documents, spanning academic papers, magazines, financial reports, and presentation slides. Benchmark data is selected from these candidates in three steps. First, to emphasize the layout structures, we annotate each document by overlaying translucent rectangles of maximally distinguishable colors onto regions corresponding to different element types. Second, we compute embeddings using ViT-Base-Patch16-224 [32] by the visual features, and apply K-means clustering (with 1,000 clusters, two documents per cluster) to identify 2,000 layout-diverse ones. Third, we rank these documents by their Element Diversity (*ED*) scores and retain the top 537 after filtering out low-quality samples (e.g., documents with blurred or ambiguous content).

Similar to prior works [12], we leverage GPT-5 [22] for generating document QA pairs. We first craft task-specific instructions and few-shot examples, and distinguish between look-up and multi-hop questions. Question types are then sampled by preset ratio (e.g., text:hybrid:hierarchy:location = 9:5:3:3), for each we prompt GPT-5 to generate similar questions following the crafted instructions and examples. The generated QA pairs are reviewed and refined by humans, where annotators prioritize rewriting defective questions into multi-hop queries requiring complex structural and content-based reasoning. All QA pairs are finalized after a second round of correctness verification.

Table 1. The Statistics of Document Datasets.

Bench	DUDE	MP-DocVQA	M3DocVQA	MMDA
ED	0.313	0.305	0.323	**0.264**
LD	1.35	0.94	1.26	**1.85**
Nodes	17.2	27.6	38.3	**49.2**
Leaves	12.6	20.8	26.2	**37.7**
Levels	3.36	3.66	**4.96**	4.23

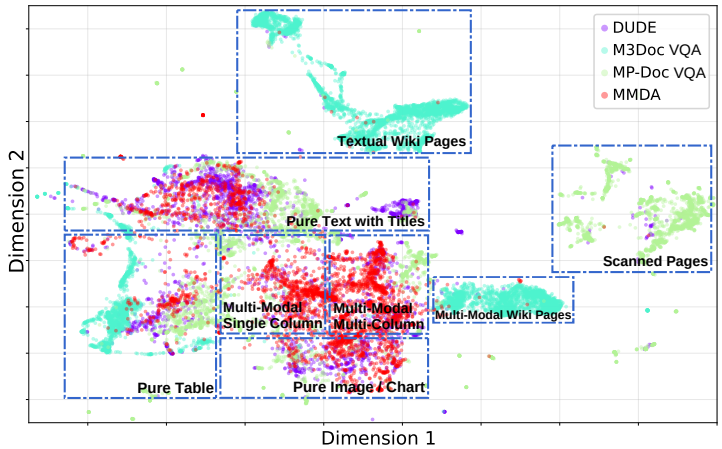


Fig. 7. Distribution of Document Layout Characteristics.

Dataset Statistics. After data preparation, the MMDA benchmark comprises 1,065 QA pairs across 537 documents. Specifically, the dataset includes 444 text-based questions, 250 hybrid questions involving both text and non-text elements, 159 hierarchy-related questions, 151 location-based questions, and 61 questions targeting formatted text information.

Dataset Comparison. Figure 7 compares the document layout distribution of DUDE, MP-DocVQA, M3DocVQA and MMDA in the embedding space. While all datasets cover basic layouts (e.g., pure text, table, image or chart), MMDA demonstrates broader coverage of complex single-/multi-column structures with multiple modality (e.g., mixed text and tables on a page). M3DocVQA and MP-DocVQA own some distinct clusters, as their Wikipedia pages and scanned pages exhibit unique styles. Other datasets include a limited number of similar document types, contributing to the distinct clusters.

Besides, we compute several dataset statistics, including Element Diversity (ED), Layout Diversity (LD), and the average number of nodes, leaves, and levels in the CCTree for each dataset. As shown in Table 1, MMDA exhibits lower ED (0.264 vs. 0.305 for MP-DocVQA), higher LD (1.85 vs. 1.35 for DUDE), and more tree nodes (49.2 vs. 38.3 for M3DocVQA), reflecting its greater element variety, more complex layout distributions, and richer content. While MMDA has slightly fewer CCTree levels than M3DocVQA, this is partly due to M3DocVQA's focus on Wikipedia documents, which feature inherently deep hierarchical structures.

Evaluation Metrics. To accurately evaluate diverse real-world questions (e.g., open-ended ones [34]), we adopt two main metrics: (1) **ACNLS**: We extend the standard ANLS [11], which uses edit distance to tolerate minor textual variations. Since verbose responses may incur low ANLS scores despite containing correct answers, we assign a score of 100% when the response contains the reference answer. (2) **AIC-Acc**: We employ LLM-based judgment to handle semantic

equivalence across varied phrasings, assigning 1 for correct answers and 0 otherwise. Similarly, responses containing the reference answer receive 100% to accommodate verbosity.

8 Experiments

8.1 Experimental Setup

Models. We employ PaddleOCR [14] for document elements parsing and Qwen3-Embedding-8B [35] for embedding search. For relatively simpler tasks such as information enrichment, summarization generation, forward search and backward verification, we locally implement Qwen2.5-VL-7B-Instruct [29]. GPT-5 is applied for more challenging tasks, including hierarchy detection, question parsing and final question answering in MoDora, as well as for the uniformly external LLM-API used in baselines.

Baselines. We compare eight typical baselines including UDOP [28], DocOWL2 [17], M3DocRAG [13], SV-RAG [12], TextRAG [16], QUEST [26], ZenDB [20] and GPT-5 [22].

Benchmarks. We evaluate MoDora and aforementioned baselines on the DUDE and MMDA, with moderate document length and diverse document layouts (see Figure 7). We separately measure their performances using ACNLS and AIC-Acc (see Section 7).

Implementation. We conduct experiments on a workstation with 2 Intel Xeon Platinum 8352V CPU (2.10 GHz), 1 TB RAM, and 8 NVIDIA RTX 4090 GPUs. We set $n_0 = 2$ and $k = 2$ for bottom-up summarization, $temperature = 1$ for LLM generation, and retrieve 3 chunks or pages via embedding search. Other baseline parameters follow the default settings in their open-source repositories [3, 4, 6].

8.2 Performance Comparison

Overall Performance. As shown in Figure 8. MoDora outperforms all baselines across all datasets and metrics, achieving 9.00% higher AIC-Acc than the best-performing baseline on DUDE, 8.46% on M3DocVQA, 5.97% on MP-DocVQA, and 16.24% on MMDA. Its superior performance can be attributed to three key factors.

First, MoDora captures both semantic and structural information from semi-structured documents, encoding them into unified component representations (Section 4). This design benefits the subsequent tree construction and analysis, enabling them to operate at a fine-grained yet complete level. In contrast, baselines do not involve this step: some (e.g., GPT-5, DocOwl2, UDOP, LayoutLMv3) process the entire document as a whole, some (e.g., SV-RAG, M3DocRAG) segment by full pages without capturing intra-page structures, and others (e.g., TextRAG, ZenDB) rely solely on textual content, ignoring layout and visual cues, limiting their ability to align and reason over structure and content of elements.

Second, MoDora leverages node indexing and summarization within the CCTree to enable hierarchical organization and multi-level abstraction, facilitating accurate and efficient retrieval from documents with multi-level nested structures. In contrast, baselines struggle with structured retrieval and reasoning. For instance, TextRAG becomes overwhelmed by excessive text due to the lack of guidance from document titles. For ZenDB, it only models titles and text while overlooking other critical elements like images and charts. Additionally, DocAgent and ZenDB ignore the fine-grained location (bounding box) of elements and fail to distinguish between main content and supplements in their hierarchical structure, leading to inferior performance compared to MoDora.

Third, the question-type-aware retrieval strategy in MoDora dynamically selects appropriate retrieval method based on question types, fully leveraging both the locational and semantic information encoded in the tree. For semantic retrieval, the LLM-guided selector, embedding search, and MLLM verifier collaborate to balance precision and recall, effectively handling complex evidence

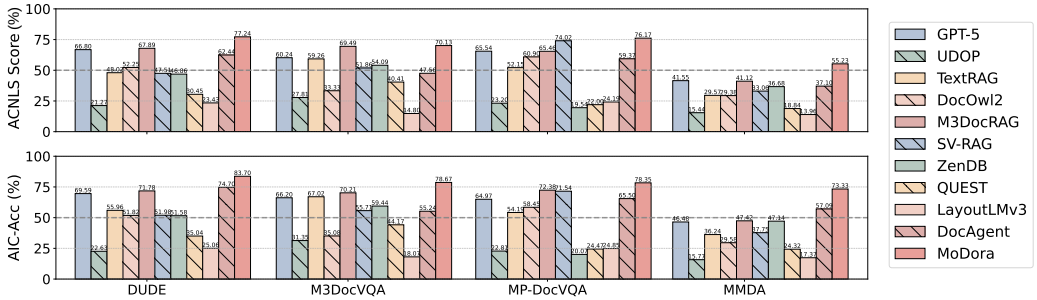


Fig. 8. Performance Comparison Between Different Baselines on Four Benchmarks.

distributions within documents (Section 6). In contrast, embedding-based methods are either too narrow, capturing only local semantics within chunks (e.g., TextRAG), or too coarse, retrieving entire pages without fine-grained discrimination (e.g., M3DocRAG and SV-RAG).

Interestingly, we observed that both SV-RAG and M3DocRAG outperform other baselines on MP-DocVQA. This is attributed to their page retrieval modules (Col-retrieval [12] for SV-RAG and ColPali [15] for M3DocRAG) being trained on DocVQA question-page pairs, which essentially correspond to the page-retrieval task in MP-DocVQA. In contrast, `MoDora`, which extracts content via OCR and then performs document modeling and analysis, achieves superior performance without specific training.

Evaluation Across Question Types. As shown in Table 2, `MoDora` performs best across all four typical question types.

(1) ***Hierarchy Questions.*** `MoDora` achieves the highest accuracy on hierarchy-based questions, demonstrating the CCTree's effectiveness in capturing document structure. By explicitly organizing elements based on heuristic aggregation rules and modeling component hierarchies, `MoDora` outperforms all baselines. While DocAgent and ZenDB rank second and third on this type, benefiting from their structural designs (XML and SHT trees), they fall significantly behind `MoDora` due to limited robustness in handling complex layouts during tree construction and retrieval.

(2) ***Textual Questions.*** For paragraph text questions, `MoDora` again outperforms all competitors. Its component-aware design ensures fine-grained representation of relevant textual blocks while avoiding unnecessary noise. DocAgent follows in performance, as its agent framework enables LLM to engage in thorough reasoning during retrieval and answering. However, its keyword-matching for semantic-based retrieval fails to fully exploit the tree structure, and does not incorporate summarization to condense detailed document content. Consequently, it still lags behind `MoDora`.

(3) ***Hybrid Questions.*** While `MoDora` maintains the top performance, hybrid questions pose a greater challenge due to the need to jointly interpret textual content alongside tables, charts and images. DocAgent ranks second because it includes tables and images in its XML tree structure outline, and allows its agent to retrieve them by table IDs or image IDs. But this approach sometimes fails due to the lack of matching IDs for the corresponding element or poor summarization of the content.

(4) ***Location Questions.*** For questions requiring location-based reasoning, `MoDora` again leads, leveraging locations in the CCTree and grid-based mapping technique to pinpoint relevant elements. GPT-5 ranks second, as its powerful visual comprehension capabilities allow it to analyze page layouts directly from full-document input. DocAgent has comparable performance to GPT-5 because its agent can retrieve document pages by number. However, it is not designed to consider the specific location (bounding box) of elements on a page, leading to failures in analysis requiring finer-grained localization, similar to using GPT-5 directly.

Table 2. AIC-Acc Performance by Question Types.

Method	Hierarchy	Text	Hybrid	Location
GPT-5	39.62	51.35	39.20	47.02
UDOP	8.81	22.97	12.40	8.61
TextRAG	20.75	50.23	37.60	18.54
DocOwl2	27.67	35.14	21.20	35.10
M3DocRAG	38.36	54.95	42.40	37.75
SV-RAG	26.42	43.92	35.60	32.45
ZenDB	52.83	60.36	36.00	35.10
QUEST	27.04	36.04	19.20	4.64
LayoutLMv3	13.21	25.23	8.40	20.53
DocAgent	55.97	68.47	53.20	46.36
MoDora	**76.73**	**79.95**	**68.00**	**68.21**

8.3 Fine-Grained Result Analysis

Analysis of Tree Construction. A well-constructed CCTree, built upon component representations, should accurately and coherently capture the content and structure of a semi-structured document. Manual observation finds the current construction method is applicable to most documents, while imperfect tree construction accounts for only about 20% of the observed analysis errors.

Suboptimal tree structures arise from OCR errors that misclassify titles as plain text, and LLM deviations in hierarchy detection. Even when certain nodes are merged or misplaced due to hierarchy inaccuracies, their metadata can still be propagated upward through the bottom-up metadata generation process and effectively utilized during tree-based retrieval, which demonstrates the robustness of the CCTree. In addition to structural errors, information enrichment may also generate incorrect descriptions of images, tables, charts, and propagate such errors into the content of tree nodes. We find that such errors occur in only ~3% of the generated results.

Analysis of Tree-based Retrieval. Tree-based retrieval process aims to identify relevant evidence nodes within the tree, balancing high precision and high recall to achieve more accurate analysis. Through manual inspection of several retrieval cases, we observe that during location-based retrieval, MoDora effectively extracts page and region evidence from the question to perform region mapping, thereby achieving high recall.

In contrast, semantics-based retrieval using the LLM-based selector and embedding search effectively identifies relevant nodes, while the MLLM verifier balances precision and recall. Manual observation reveals that the selector and embedding search recall all required evidence with a rate of 91%, and the verifier filters 84% of irrelevant nodes, though it excludes 12% of necessary evidence.

We also find that the inclusion of a few irrelevant nodes usually does not affect obtaining correct answer, whereas errors can still arise despite perfect evidence nodes (limited by analysis complexity and reasoning capability of the MLLM).

Different Backbone LLMs. To assess the impact of different backbone LLMs on MoDora, we replace GPT-5 with several mainstream LLMs for analysis, as shown in Table 3. Although ACNLS and AIC-Acc fluctuate within a certain range, the overall performance remains significantly higher than that of all baselines. These results demonstrate that the superior performance of MoDora stems from its inherent effectiveness of design rather than reliance on any particular LLM. Consequently, users can flexibly adopt different LLMs as the backbone of our framework according to their preferences and requirements.

Table 3. Performance under Different LLM Backbones.

LLM	ACNLS	AIC-Acc (%)
GPT-5	55.23	73.33
GPT-4o	57.12	70.05
Gemini-2.5-Flash	55.58	71.27
Qwen3-Max	48.02	62.63

Table 4. Performance Comparison using datasets from [6].

Dataset	Civic				Qasper	Contract
Metric	accuracy	precision	recall	f1 score	accuracy	
ZenDB	71.8	81.5	42.7	31.8	85.7	
MoDora	77.3	89.4	71.1	35.6	90.6	

Table 5. LLM-API Based Token Consumption on MMDA.

Tokens	MoDora			DocAgent	GPT-5
	Preprocess	Analysis	SUM		
Prompt	369k	6654k	7023k	52773k	771k
Completion	1084k	604k	1688k	2523k	876k

Analysis of the ACNLS metric. To verify the effectiveness of ACNLS metric, we conduct a test by using the whole-document content as the answer. It results in 43.85 on MMDA and 64.72 on DUDE. These scores are much lower than the containment score 100, and also lower than the score of MoDora shown in Figure 8. This demonstrates that the advantage of MoDora stems from the retrieval and reasoning over document content, rather than from a special metric design or the whole-document extraction.

Comparisons using ZenDB benchmarks. We further evaluate MoDora against ZenDB [6] using the same datasets and evaluation metrics reported in [6]. As shown in Table 4, MoDora consistently delivers substantially stronger performance, demonstrating robust generalizability across seven benchmarks (Civic, Qasper, Contract, DUDE, MMDA, M3DocVQA, and MP-DocVQA) and a diverse set of evaluation metrics (precision, recall, f1, ACNLS, AIC-CC).

8.4 API-Based LLM Cost Analysis

Table 5 reports the API cost of MoDora and two baselines (DocAgent, GPT-5) on MMDA. With local-model-offloading optimization (Section 8.1), MoDora consumes 7,023k prompt tokens and 1,688k completion tokens in total, averaging $0.025 per query. The pipeline requires 1, 0, and 4 API calls per query for preprocessing, tree construction, and analysis, respectively. By contrast, DocAgent incurs substantially higher costs (52,773k prompt tokens, 2,523k completion tokens; $0.09 per query) due to its multi-round interactions. Directly prompting GPT-5 with the full document is the cheapest option (771k prompt tokens, 876k completion tokens; $0.01 per query), but sacrifices accuracy. Thus, MoDora strikes a practical balance: its per-query cost remains comparable to simpler baselines while delivering over 15% higher accuracy on MMDA (Figure 8).

8.5 Ablation Study

We conduct an ablation study on four design variants of MoDora, and the results are presented in Table 6.

Table 6. Ablation Study Performance (AIC-Acc).

Methods	DUDE	MMDA Bench
Full Model	83.70	73.33
w/o Textual Evidence	70.32(-13.38)	60.56(-12.77)
w/o Locational Evidence	71.29(-12.41)	68.26(-5.07)
w/o Forward Search	78.35(-5.30)	70.89(-2.44)
w/o Tree Structure	68.37(-15.33)	57.84(-15.49)
w/o Component Construction	55.96(-27.74)	36.24(-37.09)

Without Textual Evidence. We remove textual information from the inputs to the MLLM during node verification and answer generation, leading to substantial AIC-Acc drops on DUDE (-13.38%) and MMDA (-12.77%). This highlights MLLMs' limitations in extracting key information and reasoning solely from document page images, underscoring the critical role of textual information in providing essential contextual cues.

Without Locational Evidence. We remove locational document region evidence from the MLLM inputs to assess the expressiveness of pure text. This results in performance drops of 12.41% on DUDE and 5.07% on MMDA, highlighting the importance of incorporating original document regions. They mitigate information loss during information enrichment for non-textual elements, and provide formatting cues (e.g., font size, style, color) absent in plain text.

Without Forward Search. We disable the LLM-based forward search in MoDora, relying solely on embedding search and backward verification on the tree. This leads to AIC-Acc drops of 5.3% on DUDE and 2.44% on MMDA, highlighting that retrieval guided by titles and metadata significantly improves performance.

Without Tree Structure. We evaluate the impact of hierarchical representation by performing question answering on flattened components using the same retrieval strategies. Without the tree structure, AIC-Acc drops significantly by -15.33% and -15.49% on DUDE and MoDora respectively, demonstrating the effectiveness of CCTree's structural modeling.

TextRAG. This baseline represents MoDora without three main components (i.e., component reconstruction, hierarchical representation, retrieval strategy), resulting in the most severe performance degradation, surpassing the impact of removing the hierarchical representation or retrieval strategy individually.

8.6 Case Study

In Figure 9, for questions in different types, we select representative cases to compare the performance of various methods. The reading order sequence of Doc Layout illustrates the tree structure around the evidence nodes (highlighted in gold). Based on the reading order sequence in Figure 2, the nested structure in the tree is denoted by "{ }", and evidence nodes are highlighted in gold.

We observe three main findings from the case study: (1) GPT-5 shows limited capability in information awareness and reasoning over semi-structured documents. In Cases 3–5, it retrieves only partial sets of provinces and aspects compared to the ground truth; in Case 10, it fails to integrate information from two tables for correct reasoning. (2) Embedding-based retrieval struggles to capture document hierarchies. Although TextRAG (Cases 1, 2, 6) and M3DocRAG (Cases 6–8) perform reasonably on retrieving isolated values or pages, they fail to preserve hierarchical structures critical for Cases 4 and 5, often returning irrelevant content. (3) Text-only representations result in significant information loss for non-textual elements such as tables, charts, and images. As a result, methods like TextRAG and ZenDB, which convert documents into pure text, are unable to answer most hybrid questions in Cases 6–10.

CaseID	Reading Order	Question	Type	ED	TextRAG	ZenDB	M3DocRAG	GPT-5	MoDora
1	{..., {CO1, ... CO1, ...}, ...}	What is the the Internet Boom's swansong year according to NEWS FROM KEITH ADAMS' 00?	Text	0.28	✓	✓	✗	✓	✓
2	{..., {CO1, {CO1, CO1}}, ...}	What is the willingness of residents to travel in the first quarter of 2022?	Text	0.26	✓	✗	✗	✗	✓
3	{..., {..., CO1, {..., CO1, CO1, CO1, CO1, CO1, CO1, ...}, ...}, ...}	Documents of which provinces are included in the industry dynamic?	Text	0.22	✗	✗	✓	✗	✓
4	{..., {..., CO1, {CO1, {...}, CO1, ...}, ...}, ...}	What is the the second subsection of section III about?	Hierarchy	0.30	✗	✓	✗	✗	✓
5	{..., {..., CO1, {CO1, {CO3, CO2}, CO1, CO1, CO1}, ...}, ...}	What main aspects does the Discussion include?	Hierarchy	0.29	✗	✓	✗	✗	✓
6	{..., {..., CO1, {..., CO1, ...}, ..., CO2, ...}, ...}	Is Bitmex launched earlier than South Korea's ban on Bitcoin derivatives?	Hybrid	0.24	✓	✗	✓	✓	✓
7	{..., {..., CO1, {CO1, {CO3, CO3},... }, ...}, ...}	Which month has the highest ESG monthly bond issuance volume?	Hybrid	0.23	✗	✗	✓	✓	✓
8	{..., {..., CO1, {CO1, {CO4}}, ...}, ...}	What animal is included in the figure of interest of prevention?	Hybrid	0.27	✗	✗	✓	✗	✓
9	{..., {..., CO1, {CO3, CO3}, ...}	What is the Average Income of IGA beneficiaries in Warrap State on Jan-10?	Hybrid	0.30	✗	✗	✗	✗	✓
10	{..., {..., CO1, {CO2, CO2, CO2, CO2}, ...}, ...}	What is the operating cash flow in the year with the largest operating expenses before 2022?	Hybrid	0.21	✗	✗	✗	✗	✓
11	{CO5, CO1, {CO2}, CO5, ...}	What is the text in the logo at the top of the first page?	Location	0.28	✗	✗	✗	✓	✓
12	{..., {..., CO1, {..., CO1, {CO_3}, ..., CO1, {CO_4}, ...}, ...}, ...}	How many figures are on each of the first three pages?	Location	0.25	✗	✗	✗	✗	✓

Fig. 9. Case Study on MMDA Benchmark.

9 Conclusion

In this paper, we proposed MoDora, an LLM-powered framework for semi-structured document analysis. We adopted a local-alignment aggregation strategy to convert OCR-parsed elements into layout-aware components and extract type-specific information from hierarchical and non-text elements. Then we designed a Component-Correlation Tree (CCTree) to hierarchically organize components and capture inter-component relations through bottom-up summarization. We also proposed a question-type-aware retrieval strategy combining layout-based grid partitioning and LLM-guided semantic pruning. Experimental results on public datasets and our MMDA benchmark showed that MoDora significantly outperformed state-of-the-art methods.

In the future, we plan to enhance MoDora in three main aspects. First, we will integrate native PDF structure and typographic cues not preserved by current OCR techniques to enrich the extracted information of semi-structured documents. Second, we aim to extend the CCTree to a cross-document forest for multi-document QA. Third, we will explore more advanced retrieval strategies with dynamic traversal, such as backtracking and query-driven skipping, to more effectively address potentially complex queries.

Acknowledgments

Xuanhe Zhou and Yeye He are the corresponding authors. This work was supported in part by National Key R&D Program of China (2023YFB4502400), NSF of China (62502304, U25A20437, 62525202, 62232009, 62441236, 62372296, 62432007, and U25A6024), Fundamental and Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China (No. JYB2025XDXM103), Shenzhen Project (CJGJZD20230724093403007), CCF Populus Grove Fund, ByteDance, Tencent, Ant Group through CCF-Ant Research Fund, Shanghai Jiao Tong University AI for Engineering Initiative, Shanghai Artificial Intelligence Laboratory, Alibaba Group through Alibaba Innovation Research Program, Tencent Rhino Bird Key Research Project, China Railway Science Research Institute Group Co., Ltd, Zhongguancun Lab, Huawei, and Beijing National Research Center for Information Science and Technology (BNRist).

References

[1] [n.d.]. https://pdfdrive.webs.nf.

[2] [n.d.]. https://www.sci-hub.se.

[3] 2025. *DocAgent*. https://github.com/lisun-ai/DocAgent

[4] 2025. *m3docrag*. https://github.com/bloomberg/m3docrag

[5] 2025. *PyMuPDF 1.26.7*. https://pypi.org/project/PyMuPDF/

[6] 2025. *SHTRAG*. https://github.com/Ruiying-Ma/SHTRAG

[7] 2026. *pdfplumber 0.11.9*. https://pypi.org/project/pdfplumber/

[8] Srikar Appalaraju, Bhavan Jasani, Bhargava Urala Kota, Yusheng Xie, and R Manmatha. 2021. Docformer: End-to-end transformer for document understanding. In *Proceedings of the IEEE/CVF international conference on computer vision*. 993–1003.

[9] Simran Arora, Brandon Yang, Sabri Eyuboglu, Avanika Narayan, Andrew Hojel, Immanuel Trummer, and Christopher Ré. 2023. Language Models Enable Simple Systems for Generating Structured Views of Heterogeneous Data Lakes. *Proceedings of the VLDB Endowment* 17, 2 (2023), 92–105.

[10] Yushi Bai, Shangqing Tu, Jiajie Zhang, and et alsdf. 2025. LongBench v2: Towards Deeper Understanding and Reasoning on Realistic Long-context Multitasks. In *ACL (1)*. Association for Computational Linguistics, 3639–3664.

[11] Ali Furkan Biten, Ruben Tito, Andres Mafla, Lluis Gomez, Marçal Rusinol, Ernest Valveny, CV Jawahar, and Dimosthenis Karatzas. 2019. Scene text visual question answering. In *Proceedings of the IEEE/CVF international conference on computer vision*. 4291–4301.

[12] Jian Chen, Ruiyi Zhang, Yufan Zhou, Tong Yu, Franck Dernoncourt, Jiuxiang Gu, Ryan A Rossi, Changyou Chen, and Tong Sun. 2024. SV-RAG: LoRA-Contextualizing Adaptation of MLLMs for Long Document Understanding. *arXiv preprint arXiv:2411.01106* (2024).

[13] Jaemin Cho, Debanjan Mahata, Ozan Irsoy, Yujie He, and Mohit Bansal. 2024. M3docrag: Multi-modal retrieval is what you need for multi-page multi-document understanding. *arXiv preprint arXiv:2411.04952* (2024).

[14] Cheng Cui, Ting Sun, Manhui Lin, Tingquan Gao, Yubo Zhang, Jiaxuan Liu, Xueqing Wang, Zelun Zhang, Changda Zhou, Hongen Liu, Yue Zhang, Wenyu Lv, Kui Huang, Yichao Zhang, Jing Zhang, Jun Zhang, Yi Liu, Dianhai Yu, and Yanjun Ma. 2025. PaddleOCR 3.0 Technical Report. arXiv:2507.05595 [cs.CV] https://arxiv.org/abs/2507.05595

[15] Manuel Faysse, Hugues Sibille, Tony Wu, Bilel Omrani, Gautier Viaud, Céline Hudelot, and Pierre Colombo. 2025. ColPali: Efficient Document Retrieval with Vision Language Models. arXiv:2407.01449 [cs.IR] https://arxiv.org/abs/2407.01449

[16] Nidhi Hegde, Sujoy Paul, Gagan Madan, and Gaurav Aggarwal. 2023. Analyzing the efficacy of an llm-only approach for image-based document question answering. *arXiv preprint arXiv:2309.14389* (2023).

[17] Anwen Hu, Haiyang Xu, Liang Zhang, Jiabo Ye, Ming Yan, Ji Zhang, Qin Jin, Fei Huang, and Jingren Zhou. 2024. mplug-docowl2: High-resolution compressing for ocr-free multi-page document understanding. *arXiv preprint arXiv:2409.03420* (2024).

[18] Yupan Huang, Tengchao Lv, Lei Cui, Yutong Lu, and Furu Wei. 2022. Layoutlmv3: Pre-training for document ai with unified text and image masking. In *Proceedings of the 30th ACM international conference on multimedia*. 4083–4091.

[19] Yiming Lin, Mawil Hasan, Rohan Kosalge, Alvin Cheung, and Aditya G. Parameswaran. 2025. TWIX: Automatically Reconstructing Structured Data from Templatized Documents. *CoRR* abs/2501.06659 (2025).

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

[21] Chunwei Liu, Matthew Russo, Michael Cafarella, Lei Cao, Peter Baile Chen, Zui Chen, Michael Franklin, Tim Kraska, Samuel Madden, Rana Shahout, et al. 2025. Palimpzest: Optimizing ai-powered analytics with declarative query processing. In *Proceedings of the Conference on Innovative Database Research (CIDR)*. 2.

[22] OpenAI. 2025. *Introducing GPT-5*. Technical Report. https://openai.com/index/introducing-gpt-5/

[23] Panupong Pasupat and Percy Liang. 2015. Compositional Semantic Parsing on Semi-Structured Tables. In *ACL (1)*. The Association for Computer Linguistics, 1470–1480.

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

[25] Li Sun, Liu He, Shuyue Jia, Yangfan He, and Chenyu You. 2025. Docagent: An agentic framework for multi-modal long-context document understanding. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*. 17712–17727.

[26] Zhaoze Sun, Chengliang Chai, Qiyan Deng, Kaisen Jin, Xinyu Guo, Han Han, Ye Yuan, Guoren Wang, and Lei Cao. 2025. QUEST: Query Optimization in Unstructured Document Analysis. *Proceedings of the VLDB Endowment* 18, 11 (2025), 4560–4573.

[27] Zirui Tang, Boyu Niu, Xuanhe Zhou, Boxiu Li, Wei Zhou, Jiannan Wang, Guoliang Li, Xinyi Zhang, and Fan Wu. 2026. ST-Raptor: LLM-Powered Semi-Structured Table Question Answering. *Proc. ACM Manag. Data* (2026).

[28] Zineng Tang, Ziyi Yang, Guoxin Wang, Yuwei Fang, Yang Liu, Chenguang Zhu, Michael Zeng, Cha Zhang, and Mohit Bansal. 2023. Unifying vision, text, and layout for universal document processing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 19254–19264.

[29] Qwen Team. 2025. Qwen2.5-VL. https://qwenlm.github.io/blog/qwen2.5-vl/

[30] Rubèn Tito, Dimosthenis Karatzas, and Ernest Valveny. 2023. Hierarchical multimodal transformers for multipage docvqa. *Pattern Recognition* 144 (2023), 109834.

[31] Jordy Van Landeghem, Rubèn Tito, Łukasz Borchmann, Michał Pietruszka, Pawel Joziak, Rafal Powalski, Dawid Jurkiewicz, Mickaël Coustaty, Bertrand Anckaert, Ernest Valveny, et al. 2023. Document understanding dataset and evaluation (dude). In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 19528–19540.

[32] Bichen Wu, Chenfeng Xu, Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Zhicheng Yan, Masayoshi Tomizuka, Joseph Gonzalez, Kurt Keutzer, and Peter Vajda. 2020. Visual Transformers: Token-based Image Representation and Processing for Computer Vision. arXiv:2006.03677 [cs.CV]

[33] Chening Yang, Duy-Khanh Vu, Minh-Tien Nguyen, Xuan-Quang Nguyen, Linh Nguyen, and Hung Le. 2025. Superrag: Beyond rag with layout-aware graph modeling. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 3: Industry Track)*. 544–557.

[34] Haopeng Zhang, Philip S Yu, and Jiawei Zhang. 2025. A systematic survey of text summarization: From statistical methods to large language models. *Comput. Surveys* 57, 11 (2025), 1–41.

[35] Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025. Qwen3 Embedding: Advancing Text Embedding and Reranking Through Foundation Models. *arXiv preprint arXiv:2506.05176* (2025).

Received October 2025; revised January 2026; accepted February 2026