DOCISE: IMPLICIT STRUCTURE ENCODER FOR SEMI-STRUCTURED DOCUMENT QUESTION ANSWERING

Anonymous authors
Paper under double-blind review

ABSTRACT

Semi-structured documents are ubiquitous in scientific reports, financial statements, and technical manuals. Question answering over such documents requires simultaneous understanding of text, tables, charts, and complex hierarchical layouts. Existing methods either rely on repeatedly calling large language models (LLM) for structure parsing and retrieval, leading to high costs and latency, or extracting the content of the document and losing layout and hierarchy information, sacrificing answer accuracy. To address this, we propose DocISE, a system that achieves state-of-the-art accuracy with minimal computational expense. Our approach introduces three core modules. (1) An **Implicit Structure Encoder** trained via contrastive learning and a structure consistency loss. This module jointly embeds hierarchical relationships, spatial positions, and textual content into a dense vector space to capturing document structure. (2) A **Hybrid Retrieval Pipeline** that leverages BM25, layout fingerprints, and a lightweight cross-encoder to perform high-precision retrieval drastically reducing costs and latency. (3) A **Topology-Aware Evidence Expansion** mechanism that adaptively retrieves spatially adjacent and structurally related evidence, overcoming the evidence omission common in fixed-path retrieval methods. We evaluate DocISE on four benchmarks, and comprehensive evaluations show that DocISE achieves a new state-of-the-art accuracy on the four benchmarks. Crucially, it does so with a **50×** reduction in cost and **4×** lower query latency compared to the previous state-of-the-art Method. These results demonstrate that DocISE establishes a new optimal trade-off among accuracy, cost, and speed, offering a practical and scalable path for accurate semi-structured document analysis. Source code is available at https://anonymous.4open.science/r/DocISE.

1 INTRODUCTION

Semi-structured documents, such as scientific reports, financial statements, technical manuals, and government forms, are pervasive across real-world applications. Unlike plain text or rigidly structured databases, these documents integrate diverse elements, including paragraphs, tables, charts, images, multilevel headings, sidebars, and footnotes, often arranged in complex and irregular layouts (Ke et al., 2025; Arora et al., 2023; Tang et al., 2023). Answering natural language questions over such documents is both practically valuable and technically challenging (Xu et al., 2026). For example, as illustrated in Figure 1, consider a machine learning paper (Rao et al., 2019) containing Table 2 reporting Spearman correlation coefficients for LSTM and Transformer on a fluorescence prediction task, Figure 3 showing the predicted vs. true values for both models. A question *"How much did the pretrained Transformer improve Spearman ρ on the fluorescence task, and how effective was it?"* requires the system to: (i) extract the numerical improvement from the table (e.g., from 0.22 to 0.68); (ii) interpret the figures to understand qualitative effectiveness (e.g., tighter clustering, lower variance); (iii) synthesize the quantitative and qualitative evidence; and (iv) produce a comparative answer. Such multi-step, cross-component reasoning is typical for semi-structured document analysis.

The core challenges of semi-structured document question answering can be summarized as follows. First, elements extracted are often fragmented and detached from their semantic context. A table cell or a chart title may be isolated from the surrounding explanatory text, making it difficult

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.

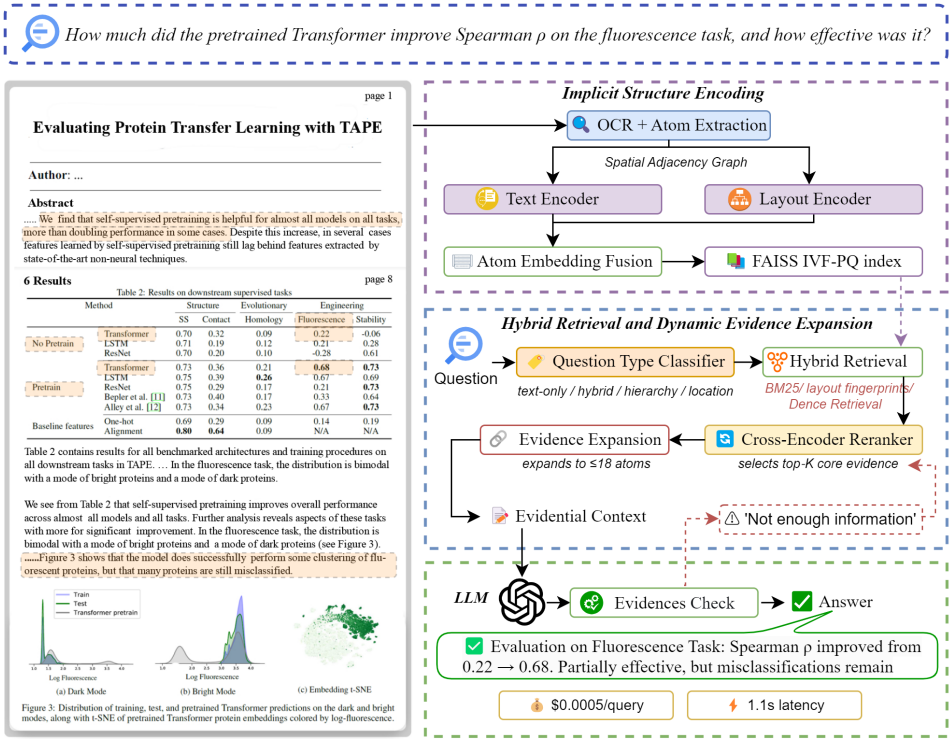

Figure 1: Overview of the DocISE pipeline and its output on an example question, illustrating the key stages and core modules of DocISE. The core modules are composed of several *small language models* (SLMs) with fewer than 100M parameters.

to interpret them in isolation (Xu et al., 2026). Second, existing methods lack effective representations to capture hierarchical structures (e.g., associating a table with nested chapter titles) and to preserve layout-specific distinctions (e.g., differentiating a sidebar from the main content). Third, answering complex questions frequently requires retrieving and aligning evidence scattered across multiple regions or pages, such as linking a descriptive paragraph to a chart located elsewhere in the document (Gong et al., 2025; Sourati et al., 2025).

A large body of research has addressed document understanding, but as shown in Figure 2, existing methods exhibit critical shortcomings when applied to semi-structured document QA. Content extraction methods transform documents into relational tables or attribute-value pairs (Sun et al., 2025b; Arora et al., 2023; Liu et al., 2025; Saad-Falcon et al., 2024; Sadia et al., 2025). While efficient for pure text, they discard layout and structural information, making them unable to answer questions that depend on spatial or hierarchical cues. Structure extraction methods (Lin et al., 2025b; 2026; Sun et al., 2025a) attempt to recover hierarchical organizations such as chapters and page mappings. However, they often fail to model non-textual elements (tables, charts) and their relations with surrounding text. End-to-end multimodal models (Huang et al., 2022; Hu et al., 2025; Achiam et al., 2023; Appalaraju et al., 2021) process documents as images and answer questions in a single pass. They suffer from high computational cost, limited context length, and poor fine-grained text understanding. In particular, they frequently overlook cross-element relationships, such as aligning a chart with its referring paragraph. Retrieval-augmented generation (RAG) methods (Chen et al., 2025; Cho et al., 2024; Lin et al., 2025a; Hegde et al., 2023; Yang et al., 2025; Cao et al., 2025) retrieve relevant chunks or page images using embeddings. They lose layout structure and often fail to perform multi-step reasoning that requires bridging information across different document regions. Tree-based structured approaches (Xu et al., 2026) represent a recent advance. They construct explicit component-correlation trees and use LLMs for node selection, summarization, and verification. While achieving state-of-the-art accuracy, they incur high cost and high latency due to multiple LLM calls during retrieval. Furthermore, the tree construction itself is fragile, and errors in hierarchy detection or OCR propagate and cause wrong or missed evidence.

We propose **DocISE**, a novel system for semi-structured document analysis that simultaneously achieves higher accuracy, lower cost, and lower latency than the current state of the art. DocISE abandons explicit tree construction and LLM-driven retrieval. Instead, it introduces three core modules: **Implicit Structure Encoding**, **Hybrid Retrieval Pipeline** and **Topology-Aware Evidence Expansion**. Finally, answer generation is performed by LLM.

This paper makes the following contributions.

- **Implicit Structure Encoding**: We propose a novel training paradigm that embeds hierarchical and layout information into dense vectors without explicit tree construction, using contrastive learning and structure consistency loss.
- **Hybrid Retrieval Pipeline**: We design a lightweight, fully local retrieval pipeline that achieves high recall and precision using only BM25, layout fingerprints, dense vectors, and a small cross-encoder, reducing cost and query latency compared to baseline methods.
- **Topology-Aware Evidence Expansion**: We introduce a mechanism that dynamically retrieves spatially and structurally related evidence atoms, overcoming the evidence omission problem of fixed-path tree retrieval.
- **State-of-the-art experimental results**: On four benchmarks, DocISE consistently outperforms prior SOTA and leads across all question types. It surpasses the previous best methods while reducing per-query cost by over 50× and latency by nearly 4×. Extensive ablations validate the contribution of each component.

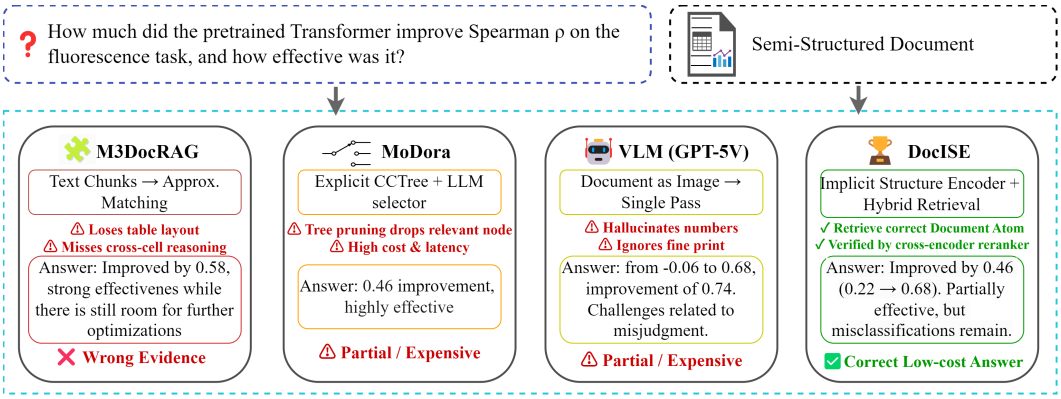

Figure 2: A comparison of different document processing methods answering a specific question about protein fluorescence. While M3DocRAG provides wrong evidence and MoDora/VLMs are costly or prone to errors, only DocISE gets the correct numerical improvement (0.46) and faithfully describes the effectiveness.

2 PRELIMINARIES

Document Atom. A semi-structured document is a visual artifact that conveys information through a mixture of textual and non-textual elements arranged in a complex layout. To enable fine-grained retrieval and reasoning, we introduce the concept of a *document atom* as the fundamental unit of information.

A document atom is a tuple $a = (type, content, bbox, page, meta)$, where: $type \in \{text, title, table_cell, chart_caption, image, supplement\}$ specifies the kind of content. *text* is a string. For textual atoms, it is the extracted text. For non-textual atoms (image), it is a short description generated by a lightweight model. $bbox = (x_1, y_1, x_2, y_2)$ is the bounding box normalized to page coordinates, where (x_1, y_1) is the top-left corner and (x_2, y_2) the bottom-right corner. *page* is the page number (starting from 1). *meta* is a set of structural hints, including: (1) $depth \in \{0, 1, 2, 3, 4, 5\}$ (estimated heading level; 0 for non-title atoms); (2) $is_title \in \{0, 1\}$;(3) *parent_embedding*: a 128-dimensional vector approximating the embedding of the atom's parent title (or zero vector if none). These meta attributes are predicted by a small auxiliary model that is trained on a small corpus of annotated documents. The model does not construct an explicit tree; it only produces per-atom estimates.

Document Representation. A document D is represented as a set of atoms $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$ together with a *spatial adjacency graph* $G_s = (V, E)$, where $V = \mathcal{A}$ and an undirected edge $(a_i, a_j) \in E$ exists if and only if: (1) $a_i.page = a_j.page$; and the bounding boxes of a_i and a_j either overlap or are within a small vertical/horizontal distance δ (we use $\delta = 0.05$ in normalized coordinates).

Semi-Structured Document QA. Let D be a semi-structured document with atom set $\mathcal{A}$ and spatial graph G_s. Let Q be a question expressed in natural language. The task is to get the answer:$D, Q) \mapsto \mathcal{R} \mapsto a$. This definition emphasizes that the system must first *retrieve* the right evidence ($\mathcal{R}$) and then *reason* over it to produce the answer. The quality of the answer thus depends critically on the recall and precision of the retrieval step.

Question Typology. To facilitate systematic evaluation, prior benchmarks (Cho et al., 2024; Van Landeghem et al., 2023; Tito et al., 2023; Xu et al., 2026) and real-world use cases identify four broad categories of questions:

- **Text-only questions** can be answered using only plain text atoms (i.e., type = text or title). Example: "What is the sample size in Experiment 2?" The answer can be extracted from a paragraph without consulting tables or charts.
- **Hybrid questions** require information from non-textual atoms, such as table cells or chart captions. Example: "Which group had the lowest feather score in week 3?" The system must locate the correct table or chart and extract the relevant cell value.
- **Hierarchy-aware questions** depend on the document's section structure. Example: "What are the two sub-topics discussed under Section 4.1?" The answer is a list of subsection titles, which requires understanding that Section 4.1 has children at a deeper depth.
- **Location-aware questions** rely on spatial position. Example: "What is the value in the bottom-left cell of the first table?" The system must interpret "bottom-left" as a region within the table's bounding box and identify the corresponding atom.

3 DocISE: Implicit Structure Encoding and Hybrid Retrieval

3.1 Offline Preprocessing

The raw document is transformed into a searchable representation. A lightweight OCR engine extracts all text fragments, table cells, chart captions, and auxiliary elements. Each extracted element becomes a document atom with its type, text, bounding box, and page number. A lightweight language model then predicts structural metadata for each atom, including a depth estimate and a title flag.

Using bounding boxes and page numbers, the system builds a spatial adjacency graph, where edges connect atoms that are on the same page and whose bounding boxes overlap or are within a small distance threshold, and the detail is in Appendix A.2. Finally, three indices are constructed: a BM25 sparse index over atom texts, a dense vector index (FAISS) over atom embeddings produced by the lightweight dual encoders, and a layout fingerprint index that encodes each page as a 6×6 grid of element type distributions. This preprocessing runs once per document and takes about two seconds for a typical ten-page document.

3.2 Implicit Structure Encoding

Dual-Encoder Architecture. As shown in Figure 1, we employ a dual-encoder (two-tower) architecture consisting of a text encoder and a layout encoder. The two encoders produce separate embeddings that are later fused into a single atom vector.

Text Encoder. The text encoder is a 6-layer Transformer encoder with 768 hidden dimensions and 12 attention heads. Its input is the concatenation of the atom's text string and a small set of structural tokens. Specifically, we prepend three special tokens: `[TYPE=type]`, `[DEPTH=d]`, and `[TITLE=is_title]`, where *type* is one of {`text`, `title`, `table_cell`, `chart_caption`, `image`, `supplement`}, d is the estimated depth (0–5), and *is_title* is 0 or 1. These tokens are

embedded and added to the standard positional embeddings. The output of the text encoder is a 768-dimensional vector h_{text}. For query encoding, we reuse the same text encoder and set the structural tokens to neutral values: `[TYPE=query]`, `[DEPTH=0]`, `[TITLE=0]`. The query is not passed through the layout encoder.

Layout Encoder. The layout encoder is also a 6-layer Transformer but takes as input a sequence of numerical features rather than text. For each atom, we extract the following layout features: normalized bounding box coordinates (x_1, y_1, x_2, y_2), page number normalized by total pages, atom type as a one-hot vector of length 6, and the relative area (width $\times$ height of the bounding box). These features are projected to 768 dimensions via a linear layer and then passed through the Transformer to produce a layout embedding h_{layout}.

Fusion. The final atom embedding is $v_a = \text{LayerNorm}(h_{\text{text}} + h_{\text{layout}})$. This additive fusion allows the model to preserve both semantic and spatial information in a single vector.

Training Objectives. The encoder is trained on a corpus of semi-structured documents with weak supervision. We use three complementary loss functions.

(1) Contrastive Learning Loss. The primary objective is to bring semantically related atoms closer in the embedding space. Positive pairs are defined as (a) atoms that belong to the same logical block, such as a table header cell and a data cell in the same column, and (b) a title atom and its directly following paragraph atom. Negative pairs are randomly sampled from different documents or distant parts of the same document. We use the standard InfoNCE loss:

$$\mathcal{L}_{\text{cont}} = -\log \frac{\exp(\text{sim}(v_a, v_{a^+})/\tau)}{\sum_{a^- \in \mathcal{N}_{\text{cont}}} \exp(\text{sim}(v_a, v_{a^-})/\tau)}$$

where sim is cosine similarity, τ is a temperature hyperparameter set to 0.07, and $\mathcal{N}_{\text{cont}}$ includes both explicitly negative atoms and in-batch negatives.

(2) Structure Consistency Loss. To capture hierarchical relationships without building an explicit tree, we introduce a loss that encourages the embedding of a title atom to be close to the embeddings of its child atoms. Specifically, for each title atom a_t and each of its children a_c, where a child is defined as an atom that appears immediately after the title, has a larger depth, and is within a certain spatial proximity, we maximize the cosine similarity between v_{a_t} and v_{a_c}. Simultaneously, we push apart v_{a_t} and embeddings of unrelated atoms. The loss is:

$$\mathcal{L}_{\text{struct}} = \sum_{(a_t, a_c) \in \mathcal{P}} (1 - \cos(v_{a_t}, v_{a_c})) + \lambda \sum_{(a_t, a_n) \in \mathcal{N}_{\text{struct}}} \max\big(0, \cos(v_{a_t}, v_{a_n}) - \delta\big)$$

where $\mathcal{P}$ is the set of title–child pairs, $\mathcal{N}_{\text{struct}}$ is a set of negative pairs consisting of a title and a random non-child atom, δ is a margin set to 0.2, and λ is a balancing weight set to 0.5.

(3) Layout Proximity Loss. To encourage spatially adjacent atoms to have similar embeddings, which helps evidence expansion, we add a weak regularization loss that minimizes the distance between atoms connected in the spatial adjacency graph G_s:

$$\mathcal{L}_{\text{layout}} = \sum_{(a_i, a_j) \in E} \| v_{a_i} - v_{a_j} \|_2^2$$

The total loss is

$$\mathcal{L} = \mathcal{L}_{\text{cont}} + \alpha \mathcal{L}_{\text{struct}} + \beta \mathcal{L}_{\text{layout}},$$

with $\alpha = 0.3$ and $\beta = 0.1$. The encoder is trained for 20 epochs on a collection of 10,000 semi-structured documents drawn from publicly available datasets: PubLayNet (scientific papers, 3,000 documents), DocBank (arXiv papers, 3,000 documents), and RVL-CDIP (forms, invoices, and reports, 4,000 documents)), and using the AdamW optimizer with a learning rate of 2×10^{-5}. Training takes about 8 hours on a single A100 GPU. The resulting atom embeddings are 768-dimensional, but we reduce them to 384 dimensions via a learned linear projection for retrieval efficiency.

Inference and Use. During preprocessing, each atom's embedding v_a is computed once and stored in a FAISS IVF-PQ index for fast approximate nearest neighbor search. At query time, the question

is encoded by the same text tower and the same projection head, with neutral structural tokens. No layout features are used. Dense retrieval then retrieves atoms with the highest cosine similarity to v_q.

3.3 Hybrid Retrieval Pipeline.

For a natural language question, the system executes a retrieval pipeline that involves no LLM calls. A **BERT-tiny classifier** first predicts the question type among four categories: text-only, hybrid, hierarchy-aware, or location-aware. The predicted question type determines which **strategy** is applied within the hybrid retrieval pipeline: (1) For text-only and hybrid questions, the system uses BM25 to retrieve a candidate set of 200 atoms, then refines it with dense vector similarity to obtain 50 candidates. (2) For hierarchy-aware questions, title atoms are identified via the structural metadata, then children are added by matching parent embeddings, which is in Appendix A.2.1. (3) For location-aware questions, spatial keywords are parsed into grid coordinates, and the layout fingerprint index directly returns atoms whose bounding boxes intersect the specified region. The top 50 atoms from this coarse retrieval are then scored by a **lightweight cross-encoder reranker** that outputs a relevance score for each (question, atom) pair. The five highest-scoring atoms serve as core evidence.

3.4 Topology-Aware Evidence Expansion.

For each core atom, the system expands the evidence set by including its spatial neighbors from the spatial adjacency graph. This expansion adds at most three atoms per core atom, yielding a final evidence set of at most 20 atoms. The texts of these atoms are concatenated in the original reading order (by page number and vertical position) to form a highly condensed context of only 500-800 tokens. The entire retrieval and expansion process takes less than 150 milliseconds(ms) on a single CPU core, as all components are local and lightweight.

Answer Generation. The final stage answers the question by calling an LLM. The LLM receives a simple prompt that includes the question and the condensed evidence context, and is instructed to answer only based on that evidence. The LLM is instructed to output "Not enough information" when it judges the evidence insufficient, then the next three highest-scoring atoms and their expanded atoms will be added to the context and calls the LLM again. In most cases, the evidences collected are sufficient to answer the question. If it is still insufficient after the added context, the final output will be "Not enough information."

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate on four public benchmarks: **MMDA** (Xu et al., 2026): 537 documents, 1,065 QA pairs with four types (text-only, hybrid, hierarchy-aware, location-aware). **MP-DocVQA** (Tito et al., 2023): Multi-page document VQA with 50,000+ questions on 12,000+ pages. **DUDE** (Van Landeghem et al., 2023): Multi-industry, multi-domain dataset with 12,000+ documents and 20,000+ questions. **M3DocVQA** (Cho et al., 2024): Multi-page, multi-document reasoning dataset with Wikipedia-sourced documents.

Baselines. We compare against 11 representative methods. *Content extraction*: QUEST (Sun et al., 2025b), UDOP (Tang et al., 2023). *Structure extraction*: ZenDB, DocAgent. *Multimodal LLMs*: LayoutLMv3, DocOwl2, GPT-5V. *RAG methods*: TextRAG, SV-RAG, M3DocRAG. *Tree-based SOTA*: MoDora.

Evaluation Metrics. **AIC-Acc** (Answer-in-Context Accuracy), where GPT-4o judges semantic equivalence to the ground truth; numeric/date answers require an exact match. **ANLS** (Average Normalized Levenshtein Similarity) with a threshold of 0.5. For retrieval analysis, we also report **Recall@k** and **MRR**(Mean Reciprocal Rank).

Implementation Details. Atom extraction uses PaddleOCR-light (Zhang et al., 2026) (confidence 0.7). The structural metadata predictor is a LayoutLMv3 variant fine-tuned on 5,000 documents from the FUNSD dataset (199 forms) and the Hierarchical Forms dataset, which do not overlap

Table 1: Overall performance comparison (ANLS and AIC-Acc %) on four benchmarks. Best results are bolded.

Method	DUDE		M3DocVQA		MP-DocVQA		MMDA	
	ANLS	AIC-Acc	ANLS	AIC-Acc	ANLS	AIC-Acc	ANLS	AIC-Acc
GPT-5V	66.80	69.59	60.24	66.20	65.54	64.97	43.55	45.53
UDOP	21.27	22.63	27.83	11.17	21.29	22.81	15.46	15.77
TextRAG	49.68	55.96	59.26	67.02	52.15	54.18	29.57	36.24
DocOwl2	50.22	11.84	32.38	35.08	60.93	58.43	9.38	19.58
M3DocRAG	22.23	51.38	69.49	70.21	55.40	72.38	41.17	47.42
SV-RAG	49.88	51.58	51.86	57.57	76.05	71.54	33.90	37.77
ZenDB	49.00	35.94	50.41	42.15	9.52	5.02	36.08	47.42
QUEST	30.45	55.96	4.88	8.07	20.01	24.40	18.84	24.32
LayoutLMv3	23.43	25.96	47.66	55.13	24.19	24.89	13.16	17.13
DocAgent	62.48	74.76	54.09	69.44	59.37	55.53	37.10	57.09
MoDora	77.24	83.70	70.13	78.67	76.17	78.35	55.23	73.33
DocISE	**80.56**	**87.52**	**76.46**	**83.68**	**78.03**	**79.86**	**67.35**	**83.92**

with the 10,000-document corpus used for the implicit structure encoder. The implicit structure encoder is trained on the same 10,000-document combined corpus described in Section 3.2. The cross-encoder reranker (50M parameters) is distilled from GPT-5.5 on 100,000 (question, atom) pairs constructed from the training splits of five public QA datasets(Appendix A.4). All indices are stored in memory (FAISS IVF-PQ, nlist=1000, pq_bytes=64). Retrieval runs on two Intel 8383C CPUs; Answer generation uses GPT-5.5. All baseline parameters follow the default settings in their open-source repositories. For multi-document QA, we do the global atomic search and aggregate by document.

4.2 Main Results

Table 1 reports overall AIC-Acc and ANLS on all four benchmarks. **DocISE consistently outperforms all baselines** across every dataset, achieving a new state-of-the-art. On MMDA, DocISE reaches **83.92%** AIC-Acc, surpassing prior SOTA MoDora (73.33%) by +10.59 points. Accuracy gains are also observed on DUDE (+3.82%), M3DocVQA (+5.01%), and MP-DocVQA (+1.51%). Three core factors jointly contribute to DocISE's superior performance.

First, implicit structure encoding replaces brittle explicit tree construction with a dense embedding space that captures hierarchical and layout information via contrastive learning and structure consistency loss. Unlike explicit methods (e.g., MoDora's Component-Correlation Tree) where a single heading detection error can propagate and cause missing or misplaced nodes, DocISE's encoder learns soft structural relationships. This makes the system **robust to OCR noise and layout variability**.

Second, zero-LLM hybrid retrieval eliminates the high cost and variance of LLM-based node selection. The pipeline combines BM25, layout fingerprints, dense vectors, and a distilled cross-encoder reranker. The cross-encoder, despite having only 50M parameters, achieves higher recall (78.4% Recall@5) than MoDora's LLM-based retrieval (71.2%) while running entirely locally,which is shown in Figure 4. This demonstrates that a **well-trained lightweight model can outperform LLM calls on this task**, at a fraction of the cost and latency. Furthermore, layout fingerprints resolve many spatial queries without dense retrieval, adding +11.8 points in ablation.

Third, dynamic evidence expansion overcomes the evidence omission problem of fixed-path tree retrieval. By expanding each core atom to include its spatial neighbors (via the spatial adjacency graph G_s) and its parent title (via parent embedding similarity), DocISE recovers cross-region evidence that tree-based methods often prune. For example, a figure caption and its descriptive paragraph located in a sidebar on a different page may be placed in disconnected branches by a fixed tree, whereas DocISE 's spatial graph naturally bridges them. Dynamic expansion adds +10.75 points overall, with even larger gains on hybrid and multi-hop questions. Unlike fixed tree paths that

require a single correct hierarchy, DocISE's expansion is query-adaptive: **it retrieves neighbors based on the question's semantics, not a predetermined structure**.

4.3 Performance Analysis

Performance by Question Type. Figure 3 shows AIC-Acc on MMDA by question type. DocISE achieves the highest accuracy across all four categories. The largest gain is on **Location questions** (+16.0 points over MoDora), where the layout fingerprint index directly maps spatial keywords to grid cells. Hybrid questions also improve substantially (+11.0 points over MoDora): they require matching table cells and chart captions, which often have sparse text; the cross-encoder reranker precisely aligns question semantics with these sparse elements, whereas MoDora's tree-based retrieval may prune such nodes. For hierarchy-aware questions (+7.8 points over MoDora), the implicit structure encoder captures parent-child relationships without explicit tree construction, avoiding propagation errors. Text-only questions improve by +7.3 points. These results confirm that DocISE 's gains are not limited to a single question type, with the largest improvement on location-aware queries.

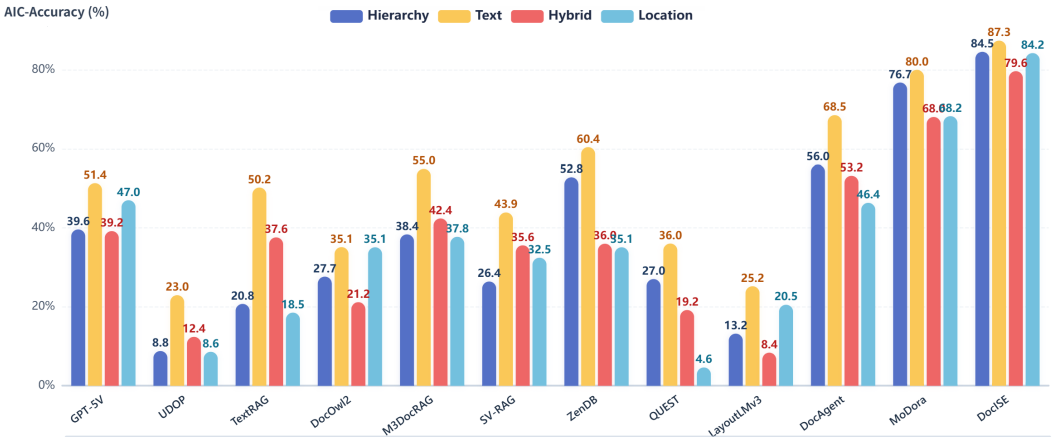

Figure 3: AIC-Acc (%) breakdown by question type on MMDA.

Retrieval Quality Analysis. To understand the source of accuracy gains, we evaluate retrieval performance independently of generation. We sample 500 questions from MMDA and compute **Recall@k** (fraction of questions where the ground-truth evidence atom is among the top k retrieved) and **MRR** (Mean Reciprocal Rank of the first correct atom). Results are shown in Figure 4-1. The cross-encoder reranker brings the largest jump (+10.6 points in Recall@5). Interestingly, DocISE 's retrieval recall (78.4% @5) surpasses MoDora's (71.2%), even though MoDora uses LLM calls for node selection.

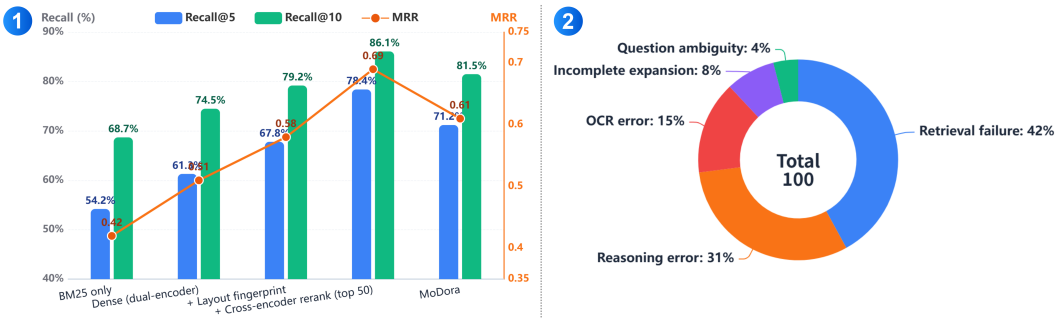

Figure 4: 1. Retrieval quality comparison on MMDA; 2. Error analysis of DocISE on MMDA

Ablation Study. Table 2 quantifies the contribution of each component. Removing the implicit structure encoder causes the largest drop (22.34 points), confirming that structure encoding is essential. The cross-encoder reranker adds +11.35 points. Dynamic evidence expansion adds +10.75

Table 2: Ablation study on MMDA (AIC-Acc %).

Configuration	AIC-Acc (%)	Drop
Full DocISE	83.92	—
w/o Implicit Structure Encoder (plain text only)	61.58	-22.34
w/o Cross-Encoder Reranker (dense retrieval only)	72.57	-11.35
w/o Dynamic Evidence Expansion (core atoms only)	73.17	-10.75
w/o Layout Encoder (text-only encoding)	73.45	-10.47
w/o Layout Fingerprint (dense for location)	72.12	-11.8
w/o Parent Embedding (no structural expansion)	76.81	-7.11
w/o Contrastive Learning (MSE only)	77.13	-6.79

points; its effect is most pronounced for questions requiring cross-region evidence (e.g., linking a figure caption to a distant paragraph). The layout encoder (spatial embedding) contributes +10.47 points, mostly from location-aware questions. Layout fingerprint contributes +11.8 points, a surprisingly large gain given its simplicity, as it resolves many spatial queries without dense retrieval.

Cost and Latency Analysis. DocISE 's per-query cost is about $0.0005 with GPT-5.5 generation, over 50× lower than MoDora ($0.025). Query latency is about 1.1 seconds vs. MoDora's 4.3 seconds, which is 4× faster. We measure retrieval latency as a function of document length. For 10-page documents, retrieval takes average 150 ms (100 cases); for 50 pages, 760 ms (sublinear due to FAISS indexing). MoDora's tree construction time grows superlinearly: from 1.2 s (10 pages) to 8.7 s (50 pages), because it builds a component tree via pairwise alignment. DocISE 's offline preprocessing also scales near-linearly: 0.2 s/page for atom extraction + 0.05 s/page for embedding.

Robustness Analysis. (1) Challenging Scenario: OCR Degradation. Simulate 20% character error rate (random substitutions, deletions, insertions). DocISE 's AIC-Acc drops from 83.92% to 73.61% (reduction of 10.31 points). MoDora drops from 73.3% to 58.1% (reduction of 15.2%). DocISE 's implicit structure encoder and dense retrieval provide robustness: embeddings are less sensitive to individual character errors than exact keyword matching used in MoDora's tree construction. (2) Challenging Scenario: Long Document. Scale documents from 10 to 50 pages by concatenating related reports. DocISE 's retrieval latency increases from 0.15 s to 0.76 s; MoDora's end-to-end latency (including LLM calls) increases from 4.3 s to 19.2 s. The primary bottleneck in MoDora is the LLM's context window, and it must process an ever-growing tree.

Error Analysis. We randomly sample 100 questions where DocISE 's final answer is incorrect. Figure 4-2 categorizes the errors. **Retrieval failure remains the largest source**, but note that 58% of errors are not retrieval-related, they are reasoning, OCR, or expansion issues. We group question type classification errors under question ambiguity. While retrieval failure is the largest single source of error, the substantial fraction of reasoning, OCR, and expansion errors indicates multiple opportunities for improvement beyond retrieval.

5 Conclusion

In this paper, we proposed DocISE, a system for semi-structured document question answering. We introduced an implicit structure encoder that jointly embeds textual content, spatial layout, and hierarchical relationships into dense vectors using contrastive learning and structure consistency loss. We designed a hybrid retrieval pipeline that combines BM25, layout fingerprints, dense vectors, and a lightweight cross-encoder, requiring no large language model calls during retrieval. We further developed a dynamic evidence expansion mechanism that retrieves spatially adjacent and structurally related atoms, overcoming the evidence omission problem of fixed-path retrieval. Experimental results on four public benchmarks demonstrate that DocISE achieves new state-of-the-art accuracy, reduces per-query cost by 50× and latency by 4× compared to existing baseline methods, and remains robust under OCR degradation and long documents. In the future, we plan to explore self-supervised pre-training for the structure encoder, and develop more compact rerankers for edge-device deployment.

6 AI USE STATEMENT

In this work, we used generative AI tools for language editing and improving the readability of the manuscript, formatting references, and assisting with literature search and summarization. We did not use generative AI tools for generating synthetic datasets, developing theoretical models or conceptual frameworks, formulating mathematical claims, assisting in the writing of proofs, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, implementing methods, translation, cleaning or reformatting datasets, supporting qualitative or thematic data analysis, or interpreting results. The remaining required-disclosure tasks are not applicable to this work. We have reviewed all AI-assisted work. For example, we checked LLM-generated text, summaries, and references for accuracy, originality, and potential plagiarism; any AI-assisted code was verified and tested by the authors. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Srikar Appalaraju, Bhavan Jasani, Bhargava Urala Kota, Yusheng Xie, and R. Manmatha. Docformer: End-to-end transformer for document understanding. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 973–983, 2021. doi: 10.1109/ICCV48922.2021.00103.

Simran Arora, Brandon Yang, Sabri Eyuboglu, Avanika Narayan, Andrew Hojel, Immanuel Trummer, and Christopher Ré. Language models enable simple systems for generating structured views of heterogeneous data lakes. *Proc. VLDB Endow.*, 17(2):92–105, October 2023. ISSN 2150-8097. doi: 10.14778/3626292.3626294. URL https://doi.org/10.14778/3626292.3626294.

Ruisheng Cao, Hanchong Zhang, Tiancheng Huang, Zhangyi Kang, Yuxin Zhang, Liangtai Sun, Hanqi Li, Yuxun Miao, Shuai Fan, Lu Chen, and Kai Yu. NeuSym-RAG: Hybrid neural symbolic retrieval with multiview structuring for PDF question answering. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6211–6239, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.311. URL https://aclanthology.org/2025.acl-long.311/.

Jian Chen, Ruiyi Zhang, Yufan Zhou, Tong Yu, Franck Dernoncourt, Jiuxiang Gu, Ryan Rossi, Changyou Chen, and Tong Sun. Sv-rag: Lora-contextualizing adaptation of mllms for long document understanding. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 4447–4462, 2025.

Jaemin Cho, Debanjan Mahata, Ozan Irsoy, Yujie He, and Mohit Bansal. M3docrag: Multi-modal retrieval is what you need for multi-page multi-document understanding, 2024. URL https://arxiv.org/abs/2411.04952.

Ziyu Gong, Chengcheng Mai, and Yihua Huang. Mhier-rag: Multi-modal rag for visual-rich document question-answering via hierarchical and multi-granularity reasoning, 2025. URL https://arxiv.org/abs/2508.00579.

Nidhi Hegde, Sujoy Paul, Gagan Madan, and Gaurav Aggarwal. Analyzing the efficacy of an llm-only approach for image-based document question answering, 2023. URL https://arxiv.org/abs/2309.14389.

Anwen Hu, Haiyang Xu, Liang Zhang, Jiabo Ye, Ming Yan, Ji Zhang, Qin Jin, Fei Huang, and Jingren Zhou. mPLUG-DocOwl2: High-resolution compressing for OCR-free multi-page document understanding. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational*

Linguistics (Volume 1: Long Papers), pp. 5817–5834, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.291. URL https://aclanthology.org/2025.acl-long.291/.

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*, MM '22, pp. 4083–4091, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450392037. doi: 10.1145/3503161.3548112. URL https://doi.org/10.1145/3503161.3548112.

Wenjun Ke, Yifan Zheng, Yining Li, Hengyuan Xu, Dong Nie, Peng Wang, and Yao He. Large language models in document intelligence: A comprehensive survey, recent advances, challenges, and future trends. *ACM Trans. Inf. Syst.*, 44(1), November 2025. ISSN 1046-8188. doi: 10.1145/3768156. URL https://doi.org/10.1145/3768156.

Teng Lin, Yizhang Zhu, Yuyu Luo, and Nan Tang. Srag: Structured retrieval-augmented generation for multi-entity question answering over wikipedia graph. *CoRR*, abs/2503.01346, March 2025a. URL https://doi.org/10.48550/arXiv.2503.01346.

Teng Lin, Yuyu Luo, and Nan Tang. Structure then query: Enabling precise analytical queries over unstructured documents, 2026. URL https://arxiv.org/abs/2608.13384.

Yiming Lin, Madelon Hulsebos, Ruiying Ma, Shreya Shankar, Sepanta Zeighami, Aditya G. Parameswaran, and Eugene Wu. Querying templatized document collections with large language models. In *2025 IEEE 41st International Conference on Data Engineering (ICDE)*, pp. 2422–2435, 2025b. doi: 10.1109/ICDE65448.2025.00183.

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

Roshan Rao, Nicholas Bhattacharya, Neil Thomas, Yan Duan, Peter Chen, John Canny, Pieter Abbeel, and Yun Song. Evaluating protein transfer learning with tape. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Jon Saad-Falcon, Joe Barrow, Alexa Siu, Ani Nenkova, Seunghyun Yoon, Ryan A. Rossi, and Franck Dernoncourt. PDFTriage: Question answering over long, structured documents. In Franck Dernoncourt, Daniel Preoţiuc-Pietro, and Anastasia Shimorina (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 153–169, Miami, Florida, US, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-industry.13. URL https://aclanthology.org/2024.emnlp-industry.13/.

Mushtari Sadia, Zhenning Yang, Yunming Xiao, Ang Chen, and Amrita Roy Chowdhury. SQUiD: Synthesizing relational databases from unstructured text. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 31987–32012, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1629. URL https://aclanthology.org/2025.emnlp-main.1629/.

Zhivar Sourati, Zheng Wang, Marianne Menglin Liu, Yazhe Hu, Mengqing Guo, Sujeeth Bharadwaj, Kyu J. Han, Tao Sheng, Sujith Ravi, Morteza Dehghani, and Dan Roth. Lad-rag: Layout-aware dynamic rag for visually-rich document understanding. *CoRR*, abs/2510.07233, October 2025. URL https://doi.org/10.48550/arXiv.2510.07233.

Li Sun, Liu He, Shuyue Jia, Yangfan He, and Chenyu You. DocAgent: An agentic framework for multi-modal long-context document understanding. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 17701–17716, Suzhou,

China, November 2025a. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.893. URL https://aclanthology.org/2025.emnlp-main.893/.

Zhaoze Sun, Chengliang Chai, Qiyan Deng, Kaisen Jin, Xinyu Guo, Han Han, Ye Yuan, Guoren Wang, and Lei Cao. Quest: Query optimization in unstructured document analysis. *Proc. VLDB Endow.*, 18(11):4560–4573, July 2025b. ISSN 2150-8097. doi: 10.14778/3749646.3749713. URL https://doi.org/10.14778/3749646.3749713.

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*, pp. 19254–19264, 2023.

Rubèn Tito, Dimosthenis Karatzas, and Ernest Valveny. Hierarchical multimodal transformers for multipage docvqa. *Pattern Recogn.*, 144(C), December 2023. ISSN 0031-3203. doi: 10.1016/j.patcog.2023.109834. URL https://doi.org/10.1016/j.patcog.2023.109834.

Jordy Van Landeghem, Rubèn Tito, Łukasz Borchmann, Michał Pietruszka, Pawel Joziak, Rafal Powalski, Dawid Jurkiewicz, Mickaël Coustaty, Bertrand Anckaert, Ernest Valveny, et al. Document understanding dataset and evaluation (dude). In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 19528–19540, 2023.

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. *Proc. ACM Manag. Data*, 4(3), May 2026. doi: 10.1145/3802089. URL https://doi.org/10.1145/3802089.

Chening Yang, Duy-Khanh Vu, Minh-Tien Nguyen, Xuan-Quang Nguyen, Linh Nguyen, and Hung Le. 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)*, pp. 544–557, 2025.

Zelun Zhang, Hongen Liu, Suyin Liang, Yubo Zhang, Yiqing Xiang, Jiaxuan Liu, Ting Sun, Manhui Lin, Yue Zhang, Changda Zhou, Tingquan Gao, Cheng Cui, Yi Liu, Dianhai Yu, and Yanjun Ma. Paddleocr-vl-1.6: Expanding the frontier of document parsing with under-optimized region refinement and progressive post-training, 2026. URL https://arxiv.org/abs/2606.03264.

A Appendices

A.1 Document Representation

A semi-structured document is a visual artifact that conveys information through a mixture of textual and non-textual elements arranged in a complex layout. Examples include scientific reports, financial statements, technical manuals, and government forms. Unlike plain text, which is a linear sequence of characters, or fully structured data, which adheres to a rigid schema, a semi-structured document contains paragraphs, titles, tables, charts, images, sidebars, headers, footers, and page numbers. These elements are not independent; they interact through spatial proximity, hierarchical grouping (e.g., sections and subsections), and cross-references (e.g., "as shown in Table 3").

The key property of an atom is that it is both *self-contained* (it can be interpreted independently) and *granular* (it does not aggregate heterogeneous content). For example, a table is not stored as a single atom; each cell is an atom. The relationship between cells in the same row or column is captured not by explicit grouping but by spatial adjacency. This design avoids the brittleness of explicit structure extraction.

The spatial graph captures local layout relationships (e.g., two cells in the same row, a caption below a chart) without imposing a hierarchical tree. It is used only during retrieval to expand evidence, as explained in later sections.

A.2 Spatial Adjacency Graph Construction

Given the set of document atoms $\mathcal{A} = \{a_1, \ldots, a_n\}$ obtained from OCR, we construct an undirected graph $G_s = (V, E)$, where $V = \mathcal{A}$ and an edge $(a_i, a_j) \in E$ is added if and only if:(1) a_i.page $=$ a_j.page; and (2) the bounding boxes of a_i and a_j satisfy either they overlap, i.e., IoU$(a_i, a_j) > 0$; or the Euclidean distance between their closest points is below a threshold $\delta = 0.05$ in normalized coordinates.

Algorithmically, for each page we sort atoms by their vertical position and scan a sliding window of size δ to avoid $O(n^2)$ comparisons. Edges are stored as adjacency lists for fast lookup during evidence expansion. The resulting graph has an average degree of 2–4, ensuring lightweight expansion while capturing meaningful local relationships, such as a caption below a chart.

A.2.1 Parent Embedding Prediction

Each document atom carries a 128-dimensional *parent embedding* that approximates the embedding of its parent title. This embedding is predicted by a lightweight auxiliary model during offline preprocessing and is used later for hierarchy-aware retrieval and topology-aware evidence expansion. The model does not construct an explicit tree; it only produces per-atom estimates.

Training Data. We collect a small corpus of annotated documents. For each document, we identify title atoms and their child atoms. A child atom is defined as an atom that appears immediately after the title, has a larger depth, and lies within a normalized vertical distance $\delta = 0.05$. For every title–child pair (a_t, a_c), the ground-truth parent embedding of a_c is set to the text embedding of a_t. Atoms without a parent title are assigned the zero vector.

Model Architecture. The parent embedding predictor is a 2-layer MLP with 256 hidden units and ReLU activation. Its input is the concatenation of:

- the atom's 768-dimensional text embedding h_{text};
- the estimated depth $d \in \{0, \ldots, 5\}$;
- the normalized bounding box coordinates (x_1, y_1, x_2, y_2);
- the page number normalized by the total number of pages.

The output is a 128-dimensional vector $p_a \in \mathbb{R}^{128}$.

Training Objective. We train the predictor with a combination of mean squared error and a contrastive loss:

$$\mathcal{L}_{\text{parent}} = \frac{1}{|\mathcal{P}|} \sum_{(a_t, a_c) \in \mathcal{P}} \|p_{a_c} - e_{a_t}\|_2^2 + \lambda_{\text{cont}} \mathcal{L}_{\text{cont}}^{\text{parent}}, \tag{1}$$

where e_{a_t} is the 128-dimensional projection of the title atom's text embedding, $\mathcal{P}$ is the set of title–child pairs, and $\mathcal{L}_{\text{cont}}^{\text{parent}}$ is an InfoNCE loss that pulls p_{a_c} close to e_{a_t} and pushes it away from randomly sampled non-parent titles. We set $\lambda_{\text{cont}} = 0.5$.

Training Details. The predictor is trained for 10 epochs using AdamW with a learning rate of 1×10^{-4} and batch size 64. Training takes about 30 minutes on 8 NVIDIA RTX 4090 GPUs. The annotated corpus contains 5,000 documents. We use the same 5,000-document subset employed for fine-tuning the structural metadata predictor. The predictor is frozen after training and used only during offline preprocessing.

Inference and Use. At preprocessing time, each atom's parent embedding is computed once and stored in the atom metadata. During hierarchy-aware retrieval, given a candidate atom a_c, we match it to the title atom whose embedding has the highest cosine similarity to p_{a_c}:

$$a_t = \arg\max_{a \in \mathcal{T}} \cos(z_{a_c}, p_a), \tag{2}$$

where $\mathcal{T}$ is the set of title atoms. If the maximum similarity is below a threshold $\theta = 0.3$, no parent is assigned.

Algorithm 1 Parent Embedding Prediction and Assignment

Require: Atom set $\mathcal{A}$, title set $\mathcal{T}$, trained predictor f_{parent}, threshold θ
Ensure: Parent embedding p_a for each atom $a \in \mathcal{A}$

Training phase:
for each annotated document D **do**
  Extract title–child pairs $\mathcal{P}$ using depth and spatial proximity
  for each pair $(a_t, a_c) \in \mathcal{P}$ **do**
    $e_{a_t} \leftarrow \textsc{Project128}(\textsc{TextEmbed}(a_t))$
    $p_{a_c} \leftarrow f_{\text{parent}}(\textsc{Features}(a_c))$
    Update f_{parent} with $\mathcal{L}_{\text{parent}}$
  end for
end for
Inference phase:
for each atom $a \in \mathcal{A}$ **do**
  $p_a \leftarrow f_{\text{parent}}(\textsc{Features}(a))$
  Store p_a in a.meta
end for
Assignment phase:
for each atom $a_c \in \mathcal{A}$ with a_c.depth > 0 **do**
  $a_t \leftarrow \arg\max_{a \in \mathcal{T}} \cos(z_{a_c}, p_a)$
  if $\cos(z_{a_c}, p_{a_t}) \geq \theta$ **then**
    Assign a_t as the parent of a_c
  else
    Assign no parent to a_c
  end if
end for

return $\{p_a \mid a \in \mathcal{A}\}$

Algorithm 1 summarizes the parent embedding prediction and assignment procedure.

Auxiliary functions.

- `TextEmbed(a)`: returns the 768-dimensional text embedding of atom a from the text tower.
- `Project128(e)`: a learned linear projection from 768 to 128 dimensions, trained jointly with the predictor.
- `Features(a)`: concatenation of h_{text}, depth, bounding box, and normalized page number.

A.3 Spatial Keyword Parsing and Layout Fingerprint

Location-aware questions require the system to map spatial expressions in natural language (e.g., *bottom-left cell of the first table*, *top-right paragraph on page 3*) to concrete document atoms. We implement this with two lightweight components: a **spatial keyword parser** and a **layout fingerprint index**. Both operate on the atom set $\mathcal{A}$ and require no LLM calls.

A.3.1 Layout Fingerprint Index

For each page p, we divide the page into a 6×6 grid. Each grid cell (r, c), where $r, c \in \{0, \ldots, 5\}$, is represented by a 6-dimensional vector $\mathbf{d}_{r,c} \in \mathbb{R}^6$, where each dimension counts the number of atoms of a given type (text, title, table_cell, chart_caption, image, supplement) whose bounding boxes intersect that cell. The vectors are normalized to sum to 1 per cell. The layout fingerprint of page p is the concatenation of all cell vectors:

$$\text{FP}(p) = \left[\mathbf{d}_{0,0}, \mathbf{d}_{0,1}, \ldots, \mathbf{d}_{5,5}\right] \in \mathbb{R}^{216}. \tag{3}$$

The layout fingerprint index $\mathcal{L}$ stores $\mathrm{FP}(p)$ for every page. It is built once during offline preprocessing and is used to quickly locate pages or regions whose element-type distribution matches a spatial query.

A.3.2 Spatial Keyword Parser

The parser takes the question q and extracts a structured spatial query:

$$(\text{page_ref},\ \text{region},\ \text{ref_object},\ \text{ordinal},\ \text{rel_pos}), \tag{4}$$

where:

- `page_ref`: page number if explicitly mentioned (e.g., "page 3"), otherwise `None`.
- `region`: one of {`top`, `bottom`, `left`, `right`, `center`, `top-left`, `top-right`, `bottom-left`, `bottom-right`, `row`, `column`, `cell`}.
- `ref_object`: reference object such as `table`, `figure`, `chart`, or `paragraph`, if present.
- `ordinal`: {`first`, `second`, `last`, `None`}.
- `rel_pos`: relative position inside the reference object, e.g., `bottom-left` for "bottom-left cell of the first table".

The parser uses a rule-based dictionary for spatial terms and dependency parsing to attach modifiers (e.g., "first table", "bottom-left cell") to their heads. If the question contains multiple spatial expressions, the parser returns the most specific one.

A.3.3 Retrieval Algorithm

Given the parsed query, the system retrieves candidate atoms as follows. If a reference object is specified, we first locate the object (e.g., the first table) and map the relative position to a sub-region of its bounding box. Otherwise, we map the region to one or more 6×6 grid cells and retrieve atoms intersecting those cells. If no atom is found, we fall back to the standard hybrid retrieval pipeline.

Algorithm 2 summarizes the procedure.

Auxiliary functions.

- `ExtractSpatialTerms(tokens)`: returns tokens matching the spatial dictionary (e.g., *top*, *bottom*, *left*, *right*, *center*, *first*, *last*, *row*, *column*, *cell*, *table*, *figure*).
- `ParseSpatialQuery(terms)`: uses dependency parsing to fill the tuple (page_ref, region, ref_object, ordinal, rel_pos).
- `FindReferenceObjects(A, type, page_ref, ordinal)`: returns atoms of the given type on the specified page, ordered by reading order; `ordinal` selects the *first*, *second*, or *last* one.
- `MapRelativePosition(rel_pos, bbox)`: maps a relative position such as `bottom-left` to a sub-rectangle of `bbox`. For example, `bottom-left` maps to the lower-left quadrant:
$$(x_1, (y_1 + y_2)/2,\ (x_1 + x_2)/2, y_2). \tag{5}$$
- `MapRegionToGrid(region)`: maps a region name to one or more 6×6 grid cells. For example, `bottom-left` maps to cells with $r \in \{3, 4, 5\}$, $c \in \{0, 1, 2\}$.
- `GridCellToBBox(r, c, page)`: converts grid cell (r, c) to a normalized bounding box on the given page:
$$\left(\frac{c}{6}, \frac{r}{6}, \frac{c+1}{6}, \frac{r+1}{6}\right). \tag{6}$$

It resolves many location-aware questions without invoking dense retrieval, which explains the large ablation gain of the layout fingerprint component in Table 2.

Algorithm 2 Spatial Keyword Parsing and Layout Fingerprint Retrieval

Require: Question q, atom set $\mathcal{A}$, layout fingerprint index $\mathcal{L}$, page dimensions
Ensure: Candidate atom set $\mathcal{R}$

tokens $\leftarrow$ TOKENIZE(q)
spatial_terms $\leftarrow$ EXTRACTSPATIALTERMS(tokens)
if spatial_terms $= \emptyset$ **then**

  return HYBRIDRETRIEVAL($q, \mathcal{A}$)
end if
(page_ref, region, ref_object, ordinal, rel_pos) $\leftarrow$ PARSESPATIALQUERY(spatial_terms)
$\mathcal{R} \leftarrow \emptyset$
if ref_object $\neq$ None **then**
  $\mathcal{O} \leftarrow$ FINDREFERENCEOBJECTS($\mathcal{A}$, ref_object, page_ref, ordinal)
  for $o \in \mathcal{O}$ **do**
    bbox_o $\leftarrow o$.bbox
    sub_region $\leftarrow$ MAPRELATIVEPOSITION(rel_pos, bbox_o)
    $\mathcal{R} \leftarrow \mathcal{R} \cup \{a \in \mathcal{A} \mid a.\text{page} = o.\text{page} \wedge a.\text{bbox} \cap \text{sub_region} \neq \emptyset\}$
  end for
else
  if page_ref $\neq$ None **then**
    $\mathcal{P} \leftarrow \{\text{page_ref}\}$
  else
    $\mathcal{P} \leftarrow$ ALLPAGES($\mathcal{A}$)
  end if
  grid_cells $\leftarrow$ MAPREGIONTOGRID(region)
  for $p \in \mathcal{P}$ **do**
    for $(r, c) \in$ grid_cells **do**
      cell_bbox $\leftarrow$ GRIDCELLTOBBOX(r, c, p)
      $\mathcal{R} \leftarrow \mathcal{R} \cup \{a \in \mathcal{A} \mid a.\text{page} = p \wedge a.\text{bbox} \cap \text{cell_bbox} \neq \emptyset\}$
    end for
  end for
end if
if $\mathcal{R} = \emptyset$ **then**

  return HYBRIDRETRIEVAL($q, \mathcal{A}$) {fallback if no spatial match}
end if

return $\mathcal{R}$

A.4 TRAINING DATASET FOR CROSS-ENCODER RERANKER

"The cross-encoder reranker (50M parameters) is distilled from GPT-5.5 on 100,000 (question, atom) pairs constructed from the training splits of five public QA datasets: TAT-QA (20,000 pairs), FinQA (20,000), WikiTableQuestions (20,000), InfographicVQA (20,000), and ChartQA (20,000). For each training question, we identify the ground-truth answer atom as a positive example and sample four negative atoms from the same document, yielding 20,000 questions × 5 pairs = 100,000 pairs. GPT-5.5 assigns a relevance score to each (question, atom) pair, and the cross-encoder is trained to regress these scores. All training documents are document-level disjoint from the evaluation sets of MMDA, DUDE, MP-DocVQA, and M3DocVQA; we verify non-overlap by comparing document hashes."

A.5 GENERATOR SENSITIVITY ANALYSIS

DocISE 's answer generation uses a single LLM call on a short context. Table 3 compares different generators using the same retrieval output.

The gap between GPT-5.5 and Qwen2.5-7B is about 11.12 points. The main difference lies in the disparity of logical reasoning, insufficient judgment and information organization abilities. How-

Table 3: Generator sensitivity analysis on MMDA.

Generator	AIC-Acc	Latency (s)	Cost per query
GPT-5.5	**83.92%**	1.1	$0.0005
GPT-4o	76.21%	1.1	$0.0001
GPT-4o-mini	74.3%	0.8	$0.00005
Qwen2.5-7B	72.8%	5.8	~
Llama-3-8B	71.5%	6.2	~

ever, Qwen2.5-7B still gets a 72.8 accuracy, and this is because DocISE 's retrieval provides a highly focused, noise-free context, reducing the burden on the generator. For cost-sensitive applications, a 7B local model is a viable choice.

A.6 Case Study: Multi-Component Reasoning

We use the fluorescence prediction task from the TAPE paper as an example. The question is: *"How much did the pretrained Transformer improve Spearman ρ on the fluorescence task, and how effective was it?"*, which is shown in Figure 2.

DocISE retrieval and answer. DocISE retrieves all related atoms:

- Atom A (table_cell):(0.22, Fluorescence, No Pretrain Transformer)
- Atom B (table_cell): (0.68, Fluorescence, Pretrain Transformer)
- Atom C (text): "We find that self-supervised pretraining is helpful for almost all models on all tasks, more than doubling performance in some case."
- Atom D (text): "Figure 3 shows that the model does successfully perform some clustering of fluorescent proteins, but that many proteins are still misclassified."

DocISE final answer: "Improved by 0.46 (from 0.22 to 0.68). Partially effective, but misclassifications remain." This answer is correct, low-cost, and faithfully grounded in the document evidence without hallucination.

This case demonstrates the synergy of DocISE 's components: the implicit structure encoder correctly associates table cells with their row and column headers; the cross-encoder ranks relevant atoms highly; dynamic expansion retrieves necessary context; and the final LLM generates an accurate answer, avoiding the pitfalls of tree pruning and visual hallucination.