HiKEY: Hierarchical Multimodal Retrieval for Open-Domain Document Question Answering

Joongmin Shin[1], **Gyuho Shim**[3], **Jeongbae Park**[1], **Jaehyung Seo**[2‡], **Heuiseok Lim**[1,3‡]

[1]Human-inspired AI Research, Korea University
[2]Computer Science and Engineering, Konkuk University
[3]Department of Computer Science and Engineering, Korea University

{tlswndals13, gjshim, insmile, limhseok}@korea.ac.kr
seojae777@konkuk.ac.kr

Abstract

Retrieval-augmented generation (RAG) for document-based open-domain question answering (ODQA) over large industrial corpora faces two core bottlenecks: routing to the correct document and combining scattered evidence. Flat text chunks and page-level images often fail to (i) identify the right document among thousands of candidates and (ii) connect multimodal evidence, such as tables and figures, within a fixed token budget. We propose **HiKEY**, a hierarchical tree-based multimodal retrieval framework that treats document hierarchy as a first-class retrieval signal. Rather than simply chunking text, **HiKEY** uses Document Hierarchical Parsing (DHP) to reconstruct a logical heterogeneous graph with explicit parent–child relations. At query time, **HiKEY** follows a hierarchical coarse-to-fine process: it first performs global routing with hierarchical indexes to prune the corpus, and then ranks sections with a multimodal fusion strategy that selects the most discriminative evidence. It finally builds a token-efficient evidence subgraph through hybrid structural–semantic packing. Experiments on ODQA benchmarks show that **HiKEY** outperforms page- and chunk-based baselines, improving retrieval recall by up to 12.9 points and end-to-end QA by up to 6.8 points.

1 Introduction

Retrieval-augmented generation (RAG) is widely used to handle large, information-dense document corpora in expert domains such as finance, law, and engineering (Lewis et al., 2021; Jeong, 2023; Ge et al., 2023). In industrial settings, documents are rarely plain text streams. They often span many pages and mix text with tables, figures, schematics, captions, and footnotes (Gong et al., 2020; Qu et al., 2025; Tito et al., 2023). Open-domain Question Answering (ODQA) further requires finding the right documents and sections from a large corpus. Thus, system performance and cost depend not only on the generator, but also on the retrieval unit and on how efficiently the system routes and aggregates evidence (Gao et al., 2024).

The dominant approach is text chunk-based RAG, which splits documents into chunks and retrieves the top-k chunks by similarity (Gong et al., 2020; Duarte et al., 2024). However, this design does not match the structure of complex industrial documents. First, answer evidence is often distributed across multiple pages or documents rather than contained in one paragraph. Second, evidence is tied to section hierarchies (e.g., 1 $\rightarrow$ 1.2 $\rightarrow$ 1.2.1) and explicit references such as "Table 2" or "Figure 3". Third, important information is often stored in non-text regions such as tables and charts (Xing et al., 2024b). Because text chunks provide only local context, they can miss global structural cues such as tables of contents and section paths. This leads to two common failures: routing failure, where the system does not find the right document or section, and incomplete answers, where relevant evidence remains disconnected (Hong et al., 2024; Shin et al., 2025).

GraphRAG methods address part of this problem by connecting chunks through graph structure (Sarthi et al., 2024; Liu et al., 2025). However, most of them still rely mainly on text links and do not fully model multimodal evidence. Another line of work embeds full-page images for multimodal RAG (Faysse et al., 2025; Cho et al., 2025; Tanaka et al., 2025). These page-level units preserve layout, but they also introduce large amounts of irrelevant content, raise token cost, and make it difficult to pack evidence from many documents into a limited context window (Ma et al., 2025).

‡Corresponding author.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 17967–17987

We argue that industrial ODQA requires more than adding modalities. It requires a retrieval unit and search procedure that use document hierarchy directly. Based on this view, we propose **HiKEY**, a framework that uses structural information as a first-class retrieval signal. **HiKEY** first reconstructs a hierarchy tree from unstructured documents and represents paragraphs, tables, and figures as multimodal evidence units with section paths. It then builds a hierarchical graph that preserves the document's logical structure.

At retrieval time, **HiKEY** follows a hierarchical coarse-to-fine strategy similar to how people read long documents. It first uses hierarchy cues to route the query to a small set of candidate documents. It then ranks sections inside those documents by scoring multimodal units (text/table/figure) with lexical, dense text-semantic, and dense visual-semantic signals. Finally, **HiKEY** assembles a budgeted evidence subgraph using hybrid packing, prioritizing sibling nodes and semantic associates so that scattered evidence can be read together.

Contributions

- **HiKEY**: We propose a hierarchical tree-based multimodal retrieval framework that upgrades document hierarchy from passive metadata to a first-class retrieval signal. By reconstructing a logical graph with DHP, **HiKEY** reduces evidence fragmentation in industrial documents and bridges flat retrieval with complex document layouts.
- **Hierarchical Coarse-to-Fine Retrieval**: We introduce a retrieval strategy that combines global routing with local fine-grained ranking. The method identifies the most discriminative evidence inside each section and enriches visual units with upper hierarchical context, making retrieval more robust to visual ambiguity and missing captions.
- **Comprehensive Experiments**: On industrial-aligned multi-page ODQA benchmarks, **HiKEY** consistently outperforms strong text-based and multimodal RAG baselines. It improves retrieval by 4.5–12.9 points and QA by 3.7–6.8 points, showing the benefit of ancestry-aware subgraph assembly under token limits.

2 Related Work

Chunking strategies and retrieval units for RAG. RAG performance is strongly affected by the retrieval unit and chunking strategy (Gao et al., 2024). Prior work ranges from fixed-length segmentation (Gong et al., 2020) to semantic or structure-aware chunking (Duarte et al., 2024; Zhao et al., 2025; Yepes et al., 2024; Verma, 2025). Still, text-centric units often miss tables and figures and provide weak global cues for corpus-level routing in ODQA, even when they use hierarchical chunking (Shin et al., 2025).

Page-level multimodal retrieval. Visual retrieval methods embed full-page images to overcome the limits of text chunking (Faysse et al., 2025; Cho et al., 2025; Tanaka et al., 2025). They preserve layout, but their granularity is coarse. A page often contains much irrelevant content, which adds noise and lowers retrieval precision. In multi-document settings, full-page inputs also increase token cost and reduce efficiency (Ma et al., 2025). Moreover, concatenating pages does not expose the logical flow of a document or the explicit links between distant pieces of evidence.

Graph-based retrieval. GraphRAG uses graphs to support contextual reasoning beyond similarity search. Some methods build knowledge graphs from entity relations (He et al., 2024), while others summarize and link text chunks for hierarchical exploration (Sarthi et al., 2024; Liu et al., 2025). However, these approaches mostly use text nodes and do not treat tables and figures as first-class retrieval units. Recent multimodal graph methods use pages as nodes (Wu et al., 2025; Jain et al., 2025), but page-level granularity still limits fine-grained evidence assembly and token efficiency.

Overall, prior methods have not fully resolved the trade-off between coverage, i.e., not missing answer documents, and efficiency, i.e., assembling complete evidence under limited resources. **HiKEY** addresses this trade-off by combining three components: (1) fine-grained multimodal units grounded in DHP-based hierarchy, (2) hierarchical coarse-to-fine retrieval that separates global routing from local ranking, and (3) ancestry-aware subgraph assembly that links scattered evidence through structural and semantic associations.

Industrial document QA and search. Enterprise document QA and search require corpus-

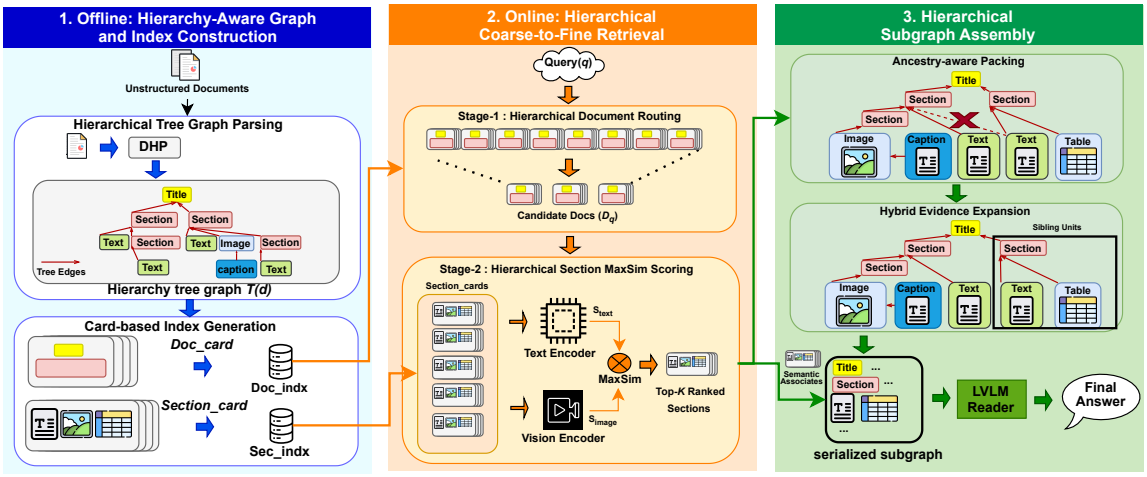

Figure 1: Overview of **HiKEY**: 1. offline hierarchy-aware graph and index construction; 2. online hierarchical coarse-to-fine retrieval with Stage-1 document routing and Stage-2 section scoring; 3. ancestry-aware evidence subgraph assembly for the LVLM reader.

level ODQA over large repositories of long PDFs. These PDFs mix text, tables, and figures, contain complex hierarchies, and must be processed under limited Reader token budgets. This is the setting targeted in this paper. Prior production-oriented studies address individual parts of this pipeline, such as long-context retrieval over enterprise-scale collections (Ma et al., 2025; Shin et al., 2025), layout- and structure-aware chunking for financial or technical documents (Yepes et al., 2024; Verma, 2025; Hong et al., 2024), and multimodal retrieval that preserves page layout (Faysse et al., 2025; Cho et al., 2025; Tanaka et al., 2025). We complement these studies by connecting corpus-level routing, block-level multimodal retrieval, and budget-aware packing in one framework. We also evaluate deployment-relevant factors directly, including strict routing (*All@K*), evidence-type and multi-hop breakdowns, and token-budget sensitivity (Tables 1, 3, and 4).

Task scope. This paper studies *corpus-level ODQA retrieval over large PDF collections*. **HiKEY** routes queries to documents, retrieves sections and units, and assembles evidence under a fixed Reader budget. We report document identification metrics (Recall/MRR/Hit/All) and end-to-end QA metrics (EM/ANLS/ROUGE-L/METEOR). We do not generate keyphrase lists or evaluate keyphrase ground truth such as keyphrase F1@K. Although our ODQA benchmarks are open-domain Wikipedia-style collections, we separately validate the DHP hierarchy backbone on scanned/pixel and multi-domain parsing benchmarks (Appendix D.1, Tab. 11), which supports robustness across document formats and domains.

3 HiKEY

HiKEY is designed to address inefficient corpus exploration and evidence fragmentation in ODQA. It follows how people search long documents: first skim the table of contents to narrow the scope, then read the relevant sections and associated tables or figures. As shown in Fig. 1, the framework has two phases: (1) offline hierarchy-aware graph and index construction, and (2) online hierarchical coarse-to-fine retrieval.

Notation. Let $\mathcal{D}$ denote the document corpus and q a user query. Each document $d \in \mathcal{D}$ is parsed into a hierarchy tree $\mathcal{T}(d)$. We define two index-card types: (i) a Doc_card x_d for the global document context and (ii) a Sec_card x_s for a section $s \in \mathcal{T}(d)$, containing multimodal units $C_s = \{c_1, c_2, \ldots, c_n\}$. Each unit c has a type $\tau(c) \in \{\text{Text, Table, Image}\}$.

3.1 Offline: Hierarchy-Aware Graph and Index Construction

To reduce the evidence fragmentation described in Sec. 1, we convert each raw document d into a structural heterogeneous graph and a searchable index.

Hierarchical Tree Graph Parsing. Conventional RAG loses global context when documents

are split into small flat chunks. We therefore use the Document Hierarchical Parsing (DHP) module from M3DocDep (Shin et al., 2026), trained on public hierarchy parsing datasets, to recover a logical hierarchy tree $\mathcal{T}(d)$ from page-level layout blocks. Each block (text, table, or figure) becomes a node, is connected by parent–child tree edges (E_{tree}), and is assigned an ancestor section path such as `Title > Section 5 > 5.3`.

Card-based Index Generation. We build hierarchical cards for retrieval:

- Doc_card (x_d): title plus section paths for global routing, $x_d = \texttt{Title}(d) \| \bigcup_{s \in \mathcal{T}(d)} \texttt{path}(s)$.
- Sec_card (x_s): a section-level card that stores its text and non-text units (tables/figures) as searchable items.

Contextual Enrichment via Graph Traversal. For visual units $c \in \{\text{Table, Image}\}$ without captions, **HiKEY** traverses the DHP tree to find a logically preceding text node. This node is stored as Upper Context $\text{ctx}(c)$ and helps reduce visual ambiguity.

3.2 Online: Hierarchical Coarse-to-Fine Retrieval

Given a query q, **HiKEY** moves through the search space from documents to sections and then to multimodal units to identify the most relevant evidence.

Stage-1: Hierarchical Document Routing. To narrow the search space, we first rank Doc_cards and select a candidate document set $\hat{\mathcal{D}}_q$ using a hybrid score:

$$S_{\text{doc}}(d, q) = \alpha \cdot \text{mm}(s_{\text{lex}}(x_d, q)) + (1-\alpha) \cdot \text{mm}(s_{\text{text}}(x_d, q))$$

where $\text{mm}(\cdot)$ is per-query min–max normalization. This stage filters irrelevant documents by using the hierarchical paths stored in Doc_cards.

Stage-2: Hierarchical Section MaxSim Scoring. Within $\hat{\mathcal{D}}_q$, we rank all Sec_cards from the candidate documents. We define a Type-Specific Unit Score $s(c, q)$ so that each modality can use an appropriate encoder:

$$s(c,q) = \begin{cases} s_{\text{hybrid}}(\text{text}(c), q) & \tau(c) = \text{Text} \\ \gamma s_{\text{vis}}(c, q) + (1-\gamma) s_{\text{hybrid}}(\text{ctx}(c), q) & \tau(c) \in \{\text{Table, Image}\} \end{cases}$$

A section's relevance is computed with MaxSim (Khattab and Zaharia, 2020) over its units: $S_{\text{sec}}(s, q) = \max_{c \in C_s} s(c, q)$. The final section score combines document-level and section-level evidence: $S_{\text{final}}(s, q) = \lambda S_{\text{doc}}(\text{doc}(s), q) + (1-\lambda) S_{\text{sec}}(s, q)$.

3.3 Hierarchical Subgraph Assembly

Finally, **HiKEY** serializes the top-K_{sec} ranked sections into a subgraph for the Reader model. To maximize information density under a fixed token budget B_{tok}, the assembly process prioritizes logical coherence and evidence connectivity.

Ancestry-aware Packing. For each selected section, we insert the Anchor Unit, i.e., the unit with the MaxSim score, together with its Governing Headers, i.e., the ancestral section titles in the DHP tree. This preserves scope and prevents the Reader from misinterpreting the evidence.

Hybrid Evidence Expansion. To connect scattered evidence without predicting an explicit cross-block link graph, we fill the remaining token budget with a hybrid packing strategy:

- Sibling Units (Structural Association): We first add tables and figures that share the same parent section as the anchor. This preserves local context and visual–text alignment inside the same subtree.
- Semantic Associates (Semantic Association): If budget remains, we add non-local visual units with high vector similarity to the anchor. This captures cross-section dependencies that are not explicit in the layout.

These two modes are complementary. Structural association recovers evidence that is co-located by layout, while semantic association recovers evidence that is topically related but placed elsewhere. Appendix E provides an algorithm-aligned schematic (Fig. 2), a compact Apollo 11 running example, and the exact Reader serialization format, showing *what the subgraph contains, how it is built, and how it is interpreted*.

4 Experiments

4.1 Experimental Settings

We evaluate **HiKEY** in a realistic corpus-level ODQA setting. All documents are indexed jointly, and the system must find the relevant documents and sections for each query. Dataset composition, metric definitions, and implementation details are provided in the Appendix.

Method	M3DocVQA				FRAMES				AVG			
	Recall	MRR	Hit	All	Recall	MRR	Hit	All	Recall	MRR	Hit	All
Text chunk-based RAG												
Page	80.6	91.3	95.0	66.4	65.4	91.9	97.1	34.4	73.0	91.6	96.1	50.4
Length chunking	82.0	90.6	95.0	69.1	65.3	92.8	96.9	33.8	73.7	91.7	96.0	51.4
LumberChunker	81.5	90.9	95.1	68.1	64.8	92.4	96.8	33.2	73.2	91.7	95.9	50.6
Meta Chunker	81.2	90.8	95.0	67.7	64.9	92.4	96.8	33.3	73.1	91.6	95.9	50.5
Structural chunking	80.6	90.5	94.8	66.9	64.0	92.1	96.6	32.4	72.3	91.3	95.7	49.7
MultiDocFusion	83.0	91.5	95.4	70.5	66.2	93.0	97.2	34.8	74.6	92.2	96.3	52.6
Text-based GraphRAG												
RAPTOR	81.8	90.9	95.0	68.4	64.9	92.0	96.6	33.3	73.4	91.5	95.8	50.9
HopRAG	82.3	91.1	95.2	69.3	65.8	92.2	96.8	34.2	74.1	91.7	96.0	51.8
Page-level multimodal RAG												
M3DocRAG	75.6	81.8	89.9	61.4	56.8	83.2	91.9	25.0	66.2	82.5	90.9	43.2
VDocRAG	77.1	83.3	90.9	63.2	58.5	84.2	92.5	26.5	67.8	83.8	91.7	44.9
Multimodal GraphRAG												
MoLoRAG	76.0	82.4	90.2	62.0	57.4	83.5	92.1	25.5	66.7	83.0	91.2	43.8
SimpleDoc	77.3	83.5	91.1	63.6	60.6	85.4	93.2	28.4	69.0	84.5	92.2	46.0
HiKEY	**84.9**	**91.8**	**96.1**	**73.6**	**73.3**	**94.2**	**97.9**	**46.2**	**79.1**	**93.0**	**97.0**	**59.9**

Table 1: Document-level retrieval results on M3DocVQA and FRAMES, averaged over $K=1,\ldots,10$, reporting Recall, MRR, Hit, and All, plus the dataset average.

Datasets. We evaluate on M3DocVQA (Cho et al., 2025) and FRAMES (Krishna et al., 2025). Together, they capture the industrial-ODQA properties discussed above: long multi-page PDFs, mixed text/table/figure evidence, fragmented multi-hop evidence, and fixed-budget serving. FRAMES is originally a Wikipedia text benchmark, so we re-render it as multi-page PDFs (FRAMES-PDF) to match our setting and to introduce visual and structural complexity. Appendix A.1 explains how each benchmark maps to enterprise requirements.

Baselines. We compare **HiKEY** with representative methods from four groups: (i) text chunk-based RAG, such as LumberChunker (Duarte et al., 2024) and MultiDocFusion (Shin et al., 2025); (ii) text-based GraphRAG, such as RAPTOR (Sarthi et al., 2024) and HopRAG (Liu et al., 2025); (iii) page-level multimodal RAG, including ColPali (Faysse et al., 2025)-based M3DocRAG (Cho et al., 2025); and (iv) page-graph multimodal GraphRAG, such as SimpleDoc (Jain et al., 2025).

Reader and Retrieval Backbones. To isolate the effect of the routing and retrieval framework, all methods use the same Reader model and decoding configuration. We use Qwen2.5-VL-7B-Instruct (Qwen Team, 2024) as the Large Vision Language Model (LVLM) Reader and fix the input context length to 16K tokens for all baselines. This reflects a practical memory constraint: page-level models can process roughly up to four pages on H100 GPUs (Cho et al., 2025). For retrieval, we use BM25 (Robertson and Zaragoza, 2009) for sparse signals, gte-Qwen2-7B-instruct (Li et al., 2023) for text embeddings, and SigLIP (Zhai et al., 2023) for visual crops such as tables and figures. Hyperparameters and library versions are detailed in Appendix F.

Protocol and Evaluation. We evaluate document identification with Recall, MRR, Hit, and All, and evaluate answer quality with EM, ANLS, ROUGE-L, and METEOR.

4.2 Experimental Results

We analyze **HiKEY** from five perspectives: (1) corpus-level localization of answer documents, (2) robustness under Top-K and token-budget constraints, (3) whether retrieval gains improve final QA, (4) performance on table/figure evidence and multi-hop queries, and (5) the contribution of each module through ablation.

4.3 Document Identification: Effect of Stage-1 Routing

Table 1 reports document identification performance averaged over $K=1,\ldots,10$. **HiKEY** performs best on both datasets, with especially large gains on FRAMES, where multiple supporting documents must be retrieved.

M3DocVQA: Stable document localization. **HiKEY** reaches 84.9 Recall, outperforming the strong MultiDocFusion baseline (83.0) by +1.9 points. On the stricter *All* metric, which requires all answer documents to be retrieved, **HiKEY** reaches 73.6, a +3.1 point gain. This shows that

Method	M3DocVQA				FRAMES				AVG			
	EM	ANLS	ROUGE-L	METEOR	EM	ANLS	ROUGE-L	METEOR	EM	ANLS	ROUGE-L	METEOR
Text chunk-based RAG												
Page	21.1	24.0	25.0	19.0	7.0	9.8	12.7	10.6	14.1	16.9	18.9	14.8
Length chunking	17.5	19.9	21.4	16.7	6.9	9.6	12.9	10.4	12.2	14.8	17.2	13.6
LumberChunker	21.4	24.2	25.6	19.5	6.8	9.4	12.5	10.2	14.1	16.8	19.1	14.9
Meta Chunker	21.3	24.1	25.5	19.4	6.8	9.4	12.5	10.2	14.1	16.8	19.0	14.8
Structural chunking	20.6	23.3	24.6	18.8	6.4	8.9	12.0	9.7	13.5	16.1	18.3	14.3
MultiDocFusion	23.0	25.9	27.3	20.6	7.8	10.8	13.9	11.6	15.4	18.4	20.6	16.1
Text-based GraphRAG												
RAPTOR	22.5	25.3	26.7	20.2	7.5	10.4	13.5	11.2	15.0	17.9	20.1	15.7
HopRAG	22.8	25.7	27.1	20.5	7.7	10.7	13.8	11.5	15.3	18.2	20.5	16.0
Page-level multimodal RAG												
M3DocRAG	24.1	27.0	28.2	21.3	4.2	5.9	9.0	6.7	14.2	16.5	18.6	14.0
VDocRAG	24.4	27.4	28.8	21.6	4.5	6.3	9.4	7.1	14.5	16.8	19.1	14.4
Multimodal GraphRAG												
MoLoRAG	25.2	28.2	30.0	22.4	4.7	6.5	9.6	7.3	15.0	17.4	19.8	14.9
SimpleDoc	25.6	28.7	30.1	22.5	4.9	6.8	9.9	7.6	15.3	17.8	20.0	15.1
HiKEY	**27.5**	**30.7**	**32.2**	**23.9**	**10.5**	**14.6**	**17.7**	**15.4**	**19.0**	**22.7**	**25.0**	**19.7**

Table 2: End-to-end QA results on M3DocVQA and FRAMES, reported with EM, ANLS, ROUGE-L, and METEOR, plus the dataset average. Higher is better for all metrics.

using hierarchy as a first-class routing signal is more effective than relying on plain text matching.

FRAMES: Resolving the routing bottleneck. The largest gain appears on FRAMES. Because FRAMES requires all dispersed supporting articles to be found, missing one document causes failure. **HiKEY** achieves 73.3 Recall (+7.1 points) and 46.2 *All*, outperforming MultiDocFusion (34.8) by +11.4 points.

These results suggest that flat and page-level retrieval often follow partial local clues and return fragmented evidence. In contrast, **HiKEY** uses global section paths to collect dispersed relevant documents with fewer omissions. Thus, **HiKEY** mitigates a common ODQA failure mode: routing failure.

Limitations of page-level multimodal retrieval. Full-page embedding methods (M3DocRAG, VDocRAG) underperform text baselines. Although they encode visual information, they lack explicit global structure for locating relevant documents in large corpora.

4.4 Top-K Sensitivity: High Recall with Few Candidates

Table 5 reports Recall as the number of retrieved documents (K) changes. While **HiKEY** is similar to baselines at R@1, the gap grows as K increases: +6.2 points at R@5 and +7.6 points at R@10.

This shows that **HiKEY** is not only selecting a good top document. It produces a stronger Top-K ranking overall. The result supports the hierarchical coarse-to-fine design: global routing provides stable candidates, and fine-grained scoring filters them precisely.

4.5 Token Budget Sensitivity: Efficiency under Limited Resources

In deployment, the input token budget is limited, so the system must pass only the most useful evidence to the Reader. Table 4 compares Avg Recall@10 under budgets from 0.5K to 2K tokens.

HiKEY performs best under every budget. Even with only 0.5K tokens, it reaches 78.8 Avg R@10, outperforming MultiDocFusion (74.3) by +4.5 points. This confirms the value of ancestry-aware packing: by selecting the Anchor Unit and its Governing Headers, **HiKEY** keeps high information density and avoids irrelevant context.

In contrast, page-level models cannot adapt flexibly to token budgets because their retrieval unit is a fixed full page.

4.6 End-to-End QA Performance

We next test whether better retrieval leads to better answers. Table 2 shows that **HiKEY** consistently improves end-to-end QA quality.

M3DocVQA. **HiKEY** achieves 27.5 EM and 30.7 ANLS, improving over the best baseline

Method	M3DocVQA								FRAMES			
	Evidence source			Doc-id hop				Overall	Doc-id hop			Overall
	Text	Table	Image	1-hop	2-hop	3-hop	4+	ANLS	2-hop	3-hop	4+	ANLS
Text chunk-based RAG												
Page	30.8	18.5	9.4	25.6	23.8	13.9	15.1	24.0	8.3	11.2	10.0	9.8
Length chunking	30.2	11.5	7.9	16.8	23.1	15.4	23.0	19.9	5.3	7.0	7.3	6.4
LumberChunker	30.9	15.5	8.8	24.8	24.5	15.8	22.6	24.2	8.0	10.8	9.6	9.4
Structural chunking	30.6	16.8	8.6	24.0	23.5	15.0	22.0	23.3	7.5	10.2	9.0	8.9
MultiDocFusion	31.4	19.8	10.8	27.0	25.8	17.2	24.4	25.9	9.2	12.5	11.0	10.8
Text-based GraphRAG												
RAPTOR	31.2	19.0	10.4	26.4	25.1	17.1	23.1	25.3	8.8	12.0	10.5	10.4
HopRAG	31.6	19.7	10.9	26.8	25.5	17.4	23.6	25.7	9.0	12.3	10.9	10.7
Page-level multimodal RAG												
M3DocRAG	31.8	22.0	13.7	31.2	25.2	12.9	2.6	27.0	8.5	7.6	0.3	5.9
VDocRAG	32.2	22.6	14.2	31.8	25.6	13.2	2.8	27.4	9.0	8.1	0.4	6.3
Multimodal GraphRAG												
MoLoRAG	32.7	22.8	14.9	32.7	26.2	11.6	5.0	28.1	9.2	8.3	0.5	6.5
SimpleDoc	33.2	23.6	15.8	33.3	26.8	12.3	6.0	28.7	9.6	8.6	0.6	6.8
HiKEY	**34.1**	**26.5**	**18.6**	**34.0**	**28.8**	**21.3**	**25.6**	**30.7**	**12.8**	**16.3**	**15.0**	**14.6**

Table 3: ANLS breakdown by evidence source (Text/Table/Image) and multi-hop difficulty (doc-id hops) on M3DocVQA and FRAMES.

Method	0.5K	0.7K	1K	2K	Avg
Text chunk-based RAG					
Page	Fixed	Fixed	Fixed	Fixed	73.3
Length chunking	73.7	73.9	74.1	76.2	74.5
LumberChunker	72.8	73.0	73.2	75.3	73.6
Meta Chunker	72.7	72.9	73.1	75.2	73.5
Structural chunking	72.0	72.2	72.3	74.5	72.8
MultiDocFusion	74.3	74.4	74.6	76.8	75.0
Text-based GraphRAG					
RAPTOR	73.0	73.2	73.4	75.5	73.8
HopRAG	73.7	73.9	74.1	76.2	74.5
Page-level multimodal RAG					
M3DocRAG	Fixed	Fixed	Fixed	Fixed	66.2
VDocRAG	Fixed	Fixed	Fixed	Fixed	67.8
Multimodal GraphRAG					
MoLoRAG	Fixed	Fixed	Fixed	Fixed	66.7
SimpleDoc	Fixed	Fixed	Fixed	Fixed	69.0
HiKEY	**78.8**	**79.0**	**79.1**	**81.3**	**79.5**

Table 4: Token-budget sensitivity averaged over M3DocVQA and FRAMES, measured by budgeted document Recall@10 after evidence packing at $B_{\text{tok}} \in \{0.5, 0.7, 1, 2\}$K. Page-level methods are marked `Fixed` because their input unit (page) cannot be resized dynamically.

Method	R@1	R@5	R@10	Avg
Text chunk-based RAG				
Page	46.3	74.3	77.3	65.9
Length chunking	47.0	76.6	80.0	67.8
LumberChunker	46.6	76.0	79.4	67.4
Meta Chunker	46.6	75.9	79.3	67.3
Structural chunking	46.1	75.1	78.5	66.6
MultiDocFusion	47.5	77.5	81.0	68.7
Text-based GraphRAG				
RAPTOR	46.8	76.2	79.6	67.5
HopRAG	47.2	77.0	80.4	68.2
Page-level multimodal RAG				
M3DocRAG	41.3	70.3	72.6	61.4
VDocRAG	42.3	72.0	74.4	62.9
Multimodal GraphRAG				
MoLoRAG	41.8	70.9	72.9	61.9
SimpleDoc	43.1	73.3	75.4	63.9
HiKEY	**47.9**	**83.7**	**88.6**	**73.4**

Table 5: Top-K sensitivity averaged over datasets, measured by document recall $R@K$ (↑). Avg is the mean over $K \in \{1, 5, 10\}$.

(SimpleDoc) by +1.9 and +2.0 points. Although the document-identification gap is modest, QA still improves. This is consistent with **HiKEY** selecting more answer-bearing units *within* retrieved documents. We do not infer this from QA scores alone; the evidence-type and hop breakdown in Sec. 4.7 and the ablations in Sec. 4.9 isolate the relevant components.

FRAMES. On FRAMES, **HiKEY** reaches 10.5 EM and 14.6 ANLS, improving over MultiDocFusion by +2.7 and +3.8 points. The gains in strict document identification (*All*) are reflected in QA performance, suggesting that for composite queries retrieving every supporting document is a key prerequisite.

4.7 Analysis by Evidence Type & Hop Count

Industrial documents contain mixed evidence types and often require multi-hop aggregation. Table 3 breaks down performance by evidence

Ablation Component	Avg R@10
Stage-1: Hierarchy Indexing Strategy	
Body-only (No hierarchy)	81.2
+ Concat Title/Header	84.6
+ Field-separated Hierarchy (Ours)	88.6
Stage-1: Routing Strategy	
Doc-only routing	87.7
Sec-only routing	76.1
Doc → Sec (Coarse-to-Fine) (Ours)	88.6
Stage-2: Multimodal Fusion Strategy	
Global Search (No routing)	81.0
Candidate Docs Only	87.9
+ Anchor Subtree (Ours)	88.6
BM25 only	87.6
+ Text dense	88.2
+ Visual dense (Full Fusion)	88.6

Table 6: Ablation study measuring averaged R@10 across key HiKEY components (higher is better). We ablate the Stage-1 hierarchy indexing strategy and routing procedure, and analyze Stage-2 design choices, including retrieval scope restrictions and multimodal fusion signals.

source and hop count.

Handling non-text evidence (Table/Image). **HiKEY** improves not only on Text, but also on Table (+2.9 points) and Image (+2.8 points) queries. This supports our choice to treat tables and figures as first-class units and enrich them with upper context through graph traversal.

Handling multi-hop queries. All methods degrade as hop count increases, but **HiKEY** degrades the least. For challenging 3-hop+ queries, **HiKEY** maintains about a 4-point advantage over baselines. This pattern is consistent with the role of hybrid subgraph assembly, which preserves local hierarchy through sibling packing and captures cross-section dependencies through semantic association.

4.8 Why Baselines Fail, Why HiKEY Helps

The strong results above do not come from a larger budget or a stronger encoder. All systems use the same LVLM Reader, corpus split, document renderings/OCR source when applicable, and 16K input budget (Sec. 4.1). Instead, each baseline family has a structural limitation in corpus-level multimodal ODQA, and **HiKEY** is designed to address it. Table 6 isolates the effects of hierarchy indexing, routing, and multimodal fusion. Table 4 and the qualitative cases in Appendix H further support the role of ancestry-aware packing. Table 7 summarizes the link between each baseline limitation, the corresponding **HiKEY** mechanism, and the supporting evidence.

Baseline family	**HiKEY mechanism (addresses the limitation)**	**Supporting evidence (in our setting)**
Chunk-based text RAG: weak document-level cues for corpus-scale routing; fragmented evidence.	`Doc_card` with a *separate* hierarchy field for Stage-1 routing.	Concat → field-separated hierarchy: 84.6 → 88.6 Avg R@10 (+4.0 points).
Text-based GraphRAG: tables/figures are not first-class retrieval units.	Sec_card with Type-Specific Unit Scoring; region-level visual embeddings.	Table/Image breakdown (Tab. 3): **HiKEY** improves over the strongest multimodal baselines; adding the visual-dense signal within Stage-2 fusion lifts Avg R@10 from 88.2 to 88.6 (Tab. 6).
Page-level multimodal RAG: whole-page units dilute evidence and saturate the Reader budget.	Block-level units + hierarchical routing; ancestry-aware packing under B_{tok}.	FRAMES strict *All*: MultiDocFusion 34.8 → **HiKEY** 46.2 (+11.4 points); **HiKEY** best at every budget (0.5K–2K, Tab. 4).
Multimodal page-graph GraphRAG: connectivity without hierarchical scope during serialization.	Ancestry-aware packing with Governing Headers + hybrid sibling/semantic expansion.	Gains hold down to the 0.5K token budget (Sec. 4.5) where page-level units no longer fit.

Table 7: Baseline-family limitations → **HiKEY** mechanism → supporting evidence in our setting. The mapping makes the source of our gains transparent in a controlled, corpus-level multimodal ODQA setting with a fixed Reader budget; these limitations have not been systematically characterized in this setting by prior work.

Appendix H gives qualitative failure cases for each baseline family. Appendix D.1 validates the DHP parser and analyzes its failure modes, clarifying how much the gains depend on each component.

4.9 Ablation Study

We ablate major components of **HiKEY** to identify where the gains come from (Table 6).

Stage-1: Hierarchy Indexing Strategy. Using hierarchy information (Title/Header) as a separate field gives a large gain over naive concatenation (84.6 → 88.6). This shows that hierarchy is most useful when it is exposed as an explicit routing signal.

Stage-1: Routing Strategy. Stepwise Doc → Sec routing outperforms Doc-only and Sec-only routing, confirming the value of the coarse-to-fine process.

Stage-2: Multimodal Fusion Strategy. Removing routing and searching the whole corpus causes a sharp drop (81.0). Restricting search to candidate documents and anchor subtrees recovers performance. The full combination of lexical, dense text, and visual signals reaches the best score (88.6), confirming the need to integrate complementary signals.

5 Conclusion

This work addresses structural bottlenecks in corpus-level ODQA over industrial multi-page documents. We identify corpus-level routing as a major practical failure mode and show that flat chunking further fragments multimodal evidence such as tables and figures. **HiKEY** mitigates these issues by using document hierarchy as a first-class retrieval signal.

HiKEY reconstructs documents into a heterogeneous graph in which multimodal blocks are enriched with DHP-derived section paths. Stage-1 uses hierarchical routing to prune the corpus and achieve high recall on complex benchmarks such as FRAMES. Stage-2 performs fine-grained retrieval inside the candidates with multimodal fusion, improving ranking quality and evidence coverage. Finally, ancestry-aware evidence packing builds a compact structural–semantic subgraph under a token budget, preserving the context needed for table- and figure-heavy queries.

Experiments on M3DocVQA and FRAMES show that **HiKEY** improves both document identification and end-to-end QA. Ablations confirm that the gains come from the combination of hierarchical indexing, coarse-to-fine retrieval, and multimodal fusion.

Limitations

Preprocessing and Indexing Overhead. Unlike simple text chunking, **HiKEY** requires offline DHP-based hierarchy reconstruction and heterogeneous graph construction. This cost is amortized at inference time, but future work should optimize the pipeline and support incremental indexing for frequently updated corpora.

Dependency on Upstream Modules. HiKEY depends on OCR and layout parsing quality. Errors in these upstream modules can produce wrong section paths and may cause routing failures. Improving robustness to noisy inputs, low-quality scans, and handwriting remains an important direction.

Structural Assumptions. HiKEY is designed for documents with explicit logical hierarchy, such as sections and headers. For weakly structured files, including flat text, receipts, or unstructured slides, hierarchical routing and ancestry-aware packing may provide less benefit than in structured PDFs.

Post-retrieval Evidence-State. Our evaluation fixes the Reader, OCR pipeline, and 16K token budget, so the reported gains are retrieval-side improvements under one delivery configuration. In our setting, retrieval recall improves by up to 12.9 points, while end-to-end QA improves by 6.8 points. Closing this gap likely requires post-retrieval improvements in OCR fidelity, evidence materialization, and reader calibration, which are outside the scope of **HiKEY**. Parser-level hierarchy metrics (F1, STEDS) and downstream retrieval/QA metrics also need not move together. Therefore, improvements in one metric should not be extrapolated to another without controlled evaluation. We leave systematic separation of retrieval gains from reader-side evidence-state factors, as well as evaluation beyond one reader family and weak-hierarchy regimes, to future work.

Acknowledgements

This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2021R1A6A1A03045425). This work was supported by the Commercialization Promotion Agency for R&D Outcomes (COMPA) grant funded by the Korea government (Ministry of Science and ICT) (2710086166). This work was supported by the Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (RS-2024-00398115, Research on the reliability and coherence of outcomes produced by Generative AI). This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2026-25481153).

References

Xiang An, Yin Xie, Kaicheng Yang, Wenkang Zhang, Xiuwei Zhao, Zheng Cheng, Yirui Wang, Songcen Xu, Changrui Chen, Chunsheng Wu, Huajie Tan, Chunyuan Li, Jing Yang, Jie Yu, Xiyao Wang, Bin Qin, Yumeng Wang, Zizhen Yan, Ziyong Feng, Ziwei Liu, Bo Li, and Jiankang Deng. 2025. Llava-onevision-1.5: Fully open framework for democratized multimodal training. *Preprint*, arXiv:2509.23661.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the ACL workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Ali Furkan Biten, Ruben Tito, Andres Mafla, Lluis Gomez, Marcal 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 (ICCV)*, pages 4291–4301.

Jaemin Cho, Debanjan Mahata, Ozan Irsoy, Yujie He, and Mohit Bansal. 2025. M3docvqa: Multi-modal multi-page multi-document understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pages 6178–6188.

André V. Duarte, João DS Marques, Miguel Graça, Miguel Freire, Lei Li, and Arlindo L. Oliveira. 2024. LumberChunker: Long-form narrative document segmentation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 6473–6486, Miami, Florida, USA. Association for Computational Linguistics.

Manuel Faysse, Hugues Sibille, Tony Wu, Bilel Omrani, Gautier Viaud, CELINE HUDELOT, and Pierre Colombo. 2025. Colpali: Efficient document retrieval with vision language models. In *The Thirteenth International Conference on Learning Representations*.

Yair Feldman and Ran El-Yaniv. 2019. Multi-hop paragraph retrieval for open-domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2296–2309, Florence, Italy.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-augmented generation for large language models: A survey. *Preprint*, arXiv:2312.10997.

J. Ge, Steve Sun, Joseph Owens, Victor Galvez, O. Gologorskaya, Jennifer C Lai, Mark J Pletcher, and Ki Lai. 2023. Development of a liver disease-specific large language model chat interface using retrieval augmented generation. *medRxiv*.

Hongyu Gong, Yelong Shen, Dian Yu, Jianshu Chen, and Dong Yu. 2020. Recurrent chunking mechanisms for long-text machine reading comprehension. pages 6751–6761.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V. Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. 2024. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. *Preprint*, arXiv:2402.07630.

Seongtae Hong, Joong Min Shin, Jaehyung Seo, Taemin Lee, Jeongbae Park, Cho Man Young, Byeongho Choi, and Heuiseok Lim. 2024. Intelligent predictive maintenance RAG framework for power plants: Enhancing QA with StyleDFS and domain specific instruction tuning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 805–820, Miami, Florida, US. Association for Computational Linguistics.

Chelsi Jain, Yiran Wu, Yifan Zeng, Jiale Liu, Shengyu Dai, Zhenwen Shao, Qingyun Wu, and Huazheng Wang. 2025. SimpleDoc: Multi-Modal document understanding with Dual-Cue page retrieval and iterative refinement. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 28398–28415, Suzhou, China. Association for Computational Linguistics.

CheonSu Jeong. 2023. A study on the implementation of generative ai services using an enterprise data-based llm application architecture. *Adv. Artif. Intell. Mach. Learn.*, 3:1588–1618.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, pages 39–48, New York, NY, USA. Association for Computing Machinery.

Satyapriya Krishna, Kalpesh Krishna, Anhad Mohananey, Steven Schwarcz, Adam Stambler, Shyam Upadhyay, and Manaal Faruqui. 2025. Fact, fetch, and reason: A unified evaluation of retrieval-augmented generation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4745–4759, Albuquerque, New Mexico. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Preprint*, arXiv:2005.11401.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023. To-

wards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Hao Liu, Zhengren Wang, Xi Chen, Zhiyu Li, Feiyu Xiong, Qinhan Yu, and Wentao Zhang. 2025. HopRAG: Multi-hop reasoning for logic-aware retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 1897–1913, Vienna, Austria. Association for Computational Linguistics.

Jiefeng Ma, Jun Du, Pengfei Hu, Zhenrong Zhang, Jianshu Zhang, Huihui Zhu, and Cong Liu. 2023. Hrdoc: dataset and baseline method toward hierarchical reconstruction of document structures. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'23/IAAI'23/EAAI'23. AAAI Press.

Yubo Ma, Jinsong Li, Yuhang Zang, Xiaobao Wu, Xiaoyi Dong, Pan Zhang, Yuhang Cao, Haodong Duan, Jiaqi Wang, Yixin Cao, and Aixin Sun. 2025. Towards storage-efficient visual document retrieval: An empirical study on reducing patch-level embeddings. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 19568–19580, Vienna, Austria. Association for Computational Linguistics.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

Birgit Pfitzmann, Christoph Auer, Michele Dolfi, Ahmed S. Nassar, and Peter Staar. 2022. Doclaynet: A large human-annotated dataset for document-layout segmentation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, pages 3743–3751, New York, NY, USA. Association for Computing Machinery.

Renyi Qu, Ruixuan Tu, and Forrest Sheng Bao. 2025. Is semantic chunking worth the computational cost? In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 2155–2177, Albuquerque, New Mexico. Association for Computational Linguistics.

Qwen Team. 2024. Qwen2.5-vl technical report. *arXiv preprint*.

Johannes Rausch, Octavio Martinez, Fabian Bissig, Ce Zhang, and Stefan Feuerriegel. 2021. Docparser: Hierarchical document structure parsing from renderings. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35:4328–4338.

Johannes Rausch, Gentiana Rashiti, Maxim Gusev, Ce Zhang, and Stefan Feuerriegel. 2023. Dsg: An end-to-end document structure generator.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. 2024. Raptor: Recursive abstractive processing for tree-organized retrieval. *Preprint*, arXiv:2401.18059.

Joongmin Shin, Chanjun Park, Jeongbae Park, Jaehyung Seo, and Heuiseok Lim. 2025. MultiDocFusion : Hierarchical and multimodal chunking pipeline for enhanced RAG on long industrial documents. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 20996–21015, Suzhou, China. Association for Computational Linguistics.

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

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

Ryota Tanaka, Taichi Iki, Taku Hasegawa, Kyosuke Nishida, Kuniko Saito, and Jun Suzuki. 2025. Vdocrag: Retrieval-augmented generation over visually-rich documents. pages 24827–24837.

Rubèn Tito, Dimosthenis Karatzas, and Ernest Valveny. 2023. Hierarchical multimodal transformers for multi-page docvqa. *Preprint*, arXiv:2212.05935.

Prashant Verma. 2025. S2 chunking: A hybrid framework for document segmentation through integrated spatial and semantic analysis. *Preprint*, arXiv:2501.05485.

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, Zhaokai Wang, Zhe Chen, Hongjie Zhang, Ganlin Yang, Haomin Wang, Qi Wei, Jinhui Yin, Wenhao Li, Erfei Cui, Guanzhou Chen, Zichen Ding, Changyao Tian, Zhenyu Wu, Jingjing Xie, Zehao Li, Bowen Yang, Yuchen Duan, Xuehui Wang, Zhi Hou, Haoran Hao, Tianyi Zhang, Songze Li, Xiangyu Zhao, Haodong Duan, Nianchen Deng, Bin Fu, Yinan He, Yi Wang, Conghui He, Botian Shi, Junjun He, Yingtong Xiong, Han Lv, Lijun Wu, Wenqi Shao, Kaipeng Zhang, Huipeng Deng, Biqing Qi, Jiaye Ge, Qipeng Guo, Wenwei Zhang, Songyang Zhang, Maosong Cao, Junyao Lin, Kexian Tang, Jianfei Gao, Haian Huang, Yuzhe Gu, Chengqi Lyu, Huanze Tang, Rui Wang, Haijun Lv, Wanli Ouyang, Limin Wang, Min Dou, Xizhou Zhu,

Tong Lu, Dahua Lin, Jifeng Dai, Weijie Su, Bowen Zhou, Kai Chen, Yu Qiao, Wenhai Wang, and Gen Luo. 2025. Internvl3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *Preprint*, arXiv:2508.18265.

Xixi Wu, Yanchao Tan, Nan Hou, Ruiyang Zhang, and Hong Cheng. 2025. MoLoRAG: Bootstrapping document understanding via multi-modal logic-aware retrieval. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 14035–14056, Suzhou, China. Association for Computational Linguistics.

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

Hangdi Xing, Changxu Cheng, et al. 2024b. Dochienet: A large and diverse dataset for document hierarchy parsing. In *EMNLP*.

Howard Yen, Tianyu Gao, Jinhyuk Lee, and Danqi Chen. 2023. Moqa: Benchmarking multi-type open-domain question answering. In *Proceedings of the Third DialDoc Workshop on Document-grounded Dialogue and Conversational Question Answering*, pages 8–29.

Antonio Jimeno Yepes, Yao You, Jan Milczek, Sebastian Laverde, and Renyu Li. 2024. Financial report chunking for effective retrieval augmented generation. *Preprint*, arXiv:2402.05131.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. 2023. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 11975–11986.

Jihao Zhao, Zhiyuan Ji, Zhaoxin Fan, Hanyu Wang, Simin Niu, Bo Tang, Feiyu Xiong, and Zhiyu Li. 2025. MoC: Mixtures of text chunking learners for retrieval-augmented generation system. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5172–5189, Vienna, Austria. Association for Computational Linguistics.

Xu Zhong, Elaheh ShafieiBavani, and Antonio Jimeno Yepes. 2020. Image-based table recognition: data, model, and evaluation. In *European Conference on Computer Vision (ECCV)*, pages 564–580. Springer.

Appendix

This appendix provides details for reproducibility and verification: (i) datasets and preprocessing, (ii) hierarchical parsing and section-path construction, (iii) hybrid packing for evidence aggregation (Sibling Units + Semantic Associates), (iv) metric definitions and evaluation protocol, including Recall/MRR/Hit/All, (v) baseline configurations, and (vi) runtime and scalability analysis.

A Datasets and Preprocessing Details

All datasets used in this paper are public research benchmarks. For RAG-based QA, we use only publicly released corpora.

A.1 ODQA Datasets

This work follows a corpus-level ODQA setting aligned with industrial requirements. Unlike standard DocVQA, where a single document is given, corpus-level ODQA jointly indexes a large PDF corpus and requires the system to find the relevant documents for each question. This directly measures routing failure, a key bottleneck in industrial RAG systems.

M3DocVQA (Cho et al., 2025). M3DocVQA is an open-domain document QA benchmark where questions target the full corpus rather than a single PDF. Some questions require multi-hop aggregation across documents, and evidence often appears in tables or figures. We index the entire M3DocVQA corpus as one retrieval space. *Industrial mapping.* This resembles enterprise search over technical manuals and product datasheets: a query is routed over a large PDF repository, and the answer may appear in a table (specifications) or a figure (schematic), not only in prose. This is why evidence-type breakdowns (Tab. 3) and token-budget sensitivity (Tab. 4) are deployment-relevant.

FRAMES (Krishna et al., 2025). FRAMES is a challenging benchmark where answers are scattered across multiple Wikipedia documents, making routing failures a major source of errors. To match our setting, we convert the original corpus into multi-page PDFs, producing FRAMES-PDF. *Industrial mapping.* This resembles cross-document compliance and due-diligence search, where the system must find every supporting document, such as filings, contracts, or regulatory exhibits. Missing one support document can invalidate the answer, which makes strict *All@K* (Tab. 1) especially important.

We rendered each Wikipedia article into PDF using a Chromium browser and the Playwright API. We preserved vector graphics, hyperlinks, and layout metadata, and prevented logical objects from being split at page boundaries. The resulting corpus has visual and structural complexity comparable to M3DocVQA and serves as a robust testbed for multimodal ODQA.

A.2 Preprocessing Pipeline

For fair comparison, all baselines and **HiKEY** use the same document renderings and OCR outputs where applicable.

Document Parsing (DP). We use a layout detection model trained on DocLayNet (Pfitzmann et al., 2022) to identify Titles, Headers, Paragraphs, Tables, Figures, and Captions. Standard post-processing with detection threshold $\tau_{\text{det}} = 0.5$ and NMS IoU threshold $\tau_{\text{nms}} = 0.5$ filters noise.

OCR. We run Tesseract (Smith, 2007) independently on each detected block region. The extracted text is lowercased, stripped of control characters, whitespace-normalized, and stored as block metadata.

B Evaluation Metrics & Protocol

This section defines the metrics and evaluation protocol used in our experiments.

B.1 Retrieval Metrics

Let $\mathcal{G}_q$ be the set of ground-truth answer documents for query q. We evaluate the retrieved set $\text{TopK}(q)$ with four metrics.

Recall@K (Manning et al., 2008). Recall measures the fraction of relevant documents retrieved:

$$\text{Recall@K}(q) = \frac{|\mathcal{G}_q \cap \text{TopK}(q)|}{|\mathcal{G}_q|}.$$

MRR@K (Yen et al., 2023). Mean Reciprocal Rank (MRR) measures ranking quality using the rank of the first relevant document. Let $\text{rank}_K(d; q)$ be the 1-indexed rank of document

[1]For non-text units such as figures and tables, **HiKEY** can attach region crops and/or multimodal embeddings while keeping each unit as a first-class node for retrieval and token-budget packing.

Method	Retrieval Unit	Modality	Field Separation	Graph	Scope
Flat RAG (page, text-only)	Page	Text	Single field (page text)	None	Corpus-level
Flat RAG (chunk, text-only)	Chunk / paragraph	Text	Single field (chunk text)	None	Corpus-level
Text GraphRAG (text-unit graph)	Chunk / paragraph	Text	Single field (text unit)	Text-unit graph	Corpus-level
Page-level multimodal RAG (Faysse et al., 2025; Cho et al., 2025; Tanaka et al., 2025)	Page	Text + Image (page embedding)	Single field (page)	None	Corpus-level
Multimodal page-graph GraphRAG (Wu et al., 2025; Jain et al., 2025)	Page	Text + Image (page embedding)	Single field (page)	Page graph	Corpus-level
HiKEY (tree-indexed multimodal GraphRAG)	Evidence unit (block-level)	Text + Image (region-based)[1]	Field-separated (`Doc_card`, section path, unit text)	Tree-based heterogeneous graph	Corpus-level

Table 8: Method taxonomy along key design axes: retrieval unit granularity, modality, field separation, graph structure, and corpus-level ODQA scope. The table shows how HiKEY differs by using block-level multimodal units and a tree-based heterogeneous graph with explicit fielded routing.

Statistic	M3DocVQA	FRAMES
Documents	3300	2500
Questions	2,441	824
Evidence source counts (with duplicates)		
Text	1,216	–
Table	1,335	–
Image	940	–
Hop distribution (by #doc_id)		
1-hop	1,096 (44.9%)	0 (0.0%)
2-hop	1,193 (48.9%)	311 (37.7%)
3-hop	114 (4.7%)	288 (35.0%)
4+ hop	38 (1.6%)	225 (27.3%)
Hop statistics		
Avg. hop	1.7	3.2
Min hop	1	2
Max hop	8	11

Table 9: Dataset statistics for M3DocVQA and FRAMES. We report the number of samples, evidence-source counts with duplicates, hop distributions based on the number of linked supporting documents (doc_id), and summary hop statistics.

d in $\mathrm{TopK}(q)$, with $\mathrm{rank}_K(d;q) = \infty$ if $d \notin \mathrm{TopK}(q)$. Then:

$$\mathrm{MRR@K}(q) = \frac{1}{\min_{d \in \mathcal{G}_q} \mathrm{rank}_K(d;q)}.$$

Hit@K (Yen et al., 2023). Hit@K checks whether at least one relevant document is retrieved:

$$\mathrm{Hit@K}(q) = \mathbb{I}\left[\mathcal{G}_q \cap \mathrm{TopK}(q) \neq \emptyset\right].$$

All@K (Feldman and El-Yaniv, 2019). Following *Potentially Perfect@K* (Feldman and El-Yaniv, 2019), All@K checks whether *all* ground-truth documents $\mathcal{G}_q$ are included in $\mathrm{TopK}(q)$. This strict metric is critical for multi-hop queries, such as FRAMES, where missing one document can cause failure:

$$\mathrm{All@K}(q) = \mathbb{I}\left[\mathcal{G}_q \subseteq \mathrm{TopK}(q)\right].$$

Reporting Protocol (Avg@1–10). To avoid selecting a single favorable K, Table 1 reports each document-retrieval metric averaged over $K \in \{1, \ldots, 10\}$.

B.2 QA Metrics

Answer quality is evaluated after standard normalization, including lowercasing and punctuation removal. We report EM (Exact Match), ANLS (Biten et al., 2019), ROUGE-L (Lin, 2004), and METEOR (Banerjee and Lavie, 2005). For ANLS, we follow the threshold-based implementation used in MP-DocVQA (Tito et al., 2023).

B.3 Token Budget Protocol

Budget Constraint. For fairness, we strictly control the input context size. Retrieved evidence units are added in ranking order until their serialized token count reaches the budget B_{tok}.

Budgeted Document Recall@10 (Table 4). Under this protocol, Table 4 reports *budgeted document Recall@10*: a ground-truth document is counted as covered only if at least one retained evidence unit from that document remains in the packed context under B_{tok}. This differs from standard Recall@10 (Table 1), which counts coverage at the document-ranking stage before packing.

Comparison with Page-level Models. Page-embedding methods such as ColPali cannot resize

their retrieval unit because each unit is a full page image. We therefore mark them as *Fixed* in budget-sensitivity analyses and compare them separately under equivalent page-count constraints, typically 4 pages $\approx$ 16K tokens.

C Baseline Implementation

All baselines use the same corpus, corpus split, Reader, and decoding settings. Each method builds its own retrieval index according to its unit and modality assumptions. Thus, the main differences come from retrieval unit, routing strategy, and modality use.

C.1 Text Chunk-based RAG

This is the standard RAG setting: flat retrieval over text units without hierarchical routing.

Page (Text-only). Treats all extracted text from a page as one retrieval unit.

Length Chunking (Gong et al., 2020). Splits text mechanically by token or character length, often breaking context.

LumberChunker (Duarte et al., 2024). Uses an LLM to detect topic shifts and set chunk boundaries based on semantic coherence.

Meta Chunker (Zhao et al., 2025). Merges sentence- or paragraph-level chunks by analyzing perplexity distributions.

Structural chunking (Yepes et al., 2024). Chunks text using layout types such as headers and paragraphs. Unlike **HiKEY**, it does not reconstruct a full section tree or treat tables and figures as first-class retrieval nodes with ancestry-aware packing.

MultiDocFusion (Shin et al., 2025). A strong hierarchy-aware baseline that preserves some structure, but lacks field-separated routing and multimodal unit fusion.

C.2 Text-based GraphRAG

These methods connect text units with graphs for multi-step exploration, but they rely only on text.

RAPTOR (Sarthi et al., 2024). Clusters and summarizes chunks to build a tree. This *induced* tree is semantic and is not tied to the document's native layout.

HopRAG (Liu et al., 2025). Expands search over a similarity graph of text units using LLM-generated pseudo-queries, but it does not natively handle tables or figures.

C.3 Page-level Multimodal RAG

These methods retrieve full-page image embeddings. They capture layout, but their retrieval unit is coarse.

M3DocRAG & VDocRAG. M3DocRAG (Cho et al., 2025) and VDocRAG (Tanaka et al., 2025) both use multimodal embeddings, but differ in scoring. M3DocRAG uses a ColPali-style late-interaction retriever with token-level page embeddings scored by MaxSim. We adapt it to ODQA by flattening all pages into one joint corpus index. VDocRAG instead uses an LVLM-based dual encoder that compresses each page image into a single dense vector, such as EOS-token pooling, for maximum inner-product search.

C.4 Multimodal GraphRAG

These methods build graphs over full pages, which limits fine-grained evidence control.

MoLoRAG (Wu et al., 2025) & SimpleDoc (Jain et al., 2025). These methods expand retrieval through page connectivity, such as similarity edges. However, page-level granularity makes it difficult to assemble a precise evidence subgraph under tight token budgets.

D Hierarchical Parsing, Section Path, and `Doc_card`

This section details the offline construction steps of **HiKEY**: hierarchical parsing (DHP), section-path extraction, and practical `Doc_card` construction.

D.1 Document Hierarchical Parsing (DHP)

Role in HiKEY. HiKEY uses an upstream Document Hierarchical Parsing (DHP) module to recover structure from raw PDFs. For a document d, DHP produces a hierarchy tree $\mathcal{T}(d)$ over layout blocks, including headings, paragraphs, captions, tables, and figures. This tree supplies section paths for Stage-1 routing and ancestry context for evidence packing. In practice, each recovered section path is serialized into the hierarchy field of Doc_cards and Sec_cards, allowing retrieval to use document-level structure rather than flat chunk similarity.

Method	Unit	Retriever Modality	Hierarchy Signal	Stage-1 Routing	Stage-2 Scope	Evidence Structure	Packing under B_{tok}
Text chunk-based RAG							
Page (Text-only)	Page	Text (OCR)	None	N	Corpus-wide	None	Fixed top-K pages (page-budget)
Length chunking (Gong et al., 2020)	Chunk	Text (OCR)	None	N	Corpus-wide	None	Greedy top-k chunks
LumberChunker (Duarte et al., 2024)	Chunk	Text (OCR)	None	N	Corpus-wide	None	Greedy top-k chunks
Meta Chunker (Zhao et al., 2025)	Chunk	Text (OCR)	None	N	Corpus-wide	None	Greedy top-k chunks
Structural chunking (Yepes et al., 2024)	Chunk	Text (OCR)	Weak (layout-based boundaries)	N	Corpus-wide	None	Greedy top-k chunks
MultiDocFusion (Shin et al., 2025)	Hier. chunk	Text (OCR)	Native / structure-aware	N (no `Hierarchy Field`)	Corpus-wide	No explicit graph (chunk hierarchy only)	Greedy top-k chunks (no sub-graph assembly)
Text-based GraphRAG							
RAPTOR (Sarthi et al., 2024)	Chunk + Summary	Text	*Induced* (summary tree)	N	Corpus-wide	Recursive summary tree	Greedy top-k nodes
HopRAG (Liu et al., 2025)	Chunk	Text	None	N	Corpus-wide	Chunk graph (similarity / hop expansion)	Greedy top-k chunks
Page-level multimodal RAG							
M3DocRAG (Cho et al., 2025)	Page	Page-level MM	None	N	Corpus-wide	None	Fixed top-K pages (page-budget)
VDocRAG (Tanaka et al., 2025)	Page	Page-level MM	None	N	Corpus-wide	None	Fixed top-K pages (page-budget)
Multimodal GraphRAG							
MoLoRAG (Wu et al., 2025)	Page	Page-level MM	None	N	Corpus-wide	Page graph expansion	Fixed top-K pages (page-budget)
SimpleDoc (Jain et al., 2025)	Page	Page-level MM	None	N	Corpus-wide	Page-level graph	Fixed top-K pages (page-budget)
HiKEY	Fine-grained MM unit	Text + Visual Crop	Native (DHP path)	Y: Hierarchy Routing	Cand. docs + Anchor sub-tree	Heterogeneous Tree Graph (tree edges only)	Ancestry-aware hybrid packing (Sibling + Semantic)

Table 10: Qualitative comparison of retrieval framework design choices. We compare methods by retrieval unit, modality, hierarchy usage, Stage-1 routing, Stage-2 search scope, evidence structure, and packing under a token budget B_{tok}.

Method	**HRDoc-S F1/STEDS**	**HRDoc-H F1/STEDS**	**DocHieNet F1/STEDS**	**Avg F1/STEDS**
Generic LVLM baselines (zero-shot image understanding)				
LLaVA-OneVision1.5 (An et al., 2025)	27.61/12.93	26.30/18.21	17.78/ 8.57	23.90/13.24
InternVL3.5 (Wang et al., 2025)	28.40/14.47	27.57/19.98	18.18/ 9.60	24.72/14.68
Qwen2.5-VL (Qwen Team, 2024)	28.41/14.51	27.57/19.99	18.20/ 9.62	24.73/14.71
Structured hierarchy parsers (trained for tree reconstruction)				
DocParser (Rausch et al., 2021)	47.09/31.03	35.41/27.15	10.68/ 4.31	31.06/20.83
DSG (Rausch et al., 2023)	48.43/32.13	36.42/27.69	26.71/19.45	37.19/26.42
DSPS (Ma et al., 2023)	65.27/59.57	54.06/38.41	35.61/23.81	51.65/40.60
DSHP-LLM (Shin et al., 2025)	44.90/29.52	61.29/51.34	64.29/53.49	56.83/44.78
Qwen2.5-VL–DHP–SFT	50.97/46.75	43.05/41.02	42.85/40.39	45.62/42.72
M3DocDep (DHP used in HiKEY)	**82.87/76.52**	**77.75/71.65**	**76.01/70.83**	**78.88/72.99**

Table 11: Standalone validation of the DHP backbone used in HiKEY, adapted from M3DocDep (Shin et al., 2026). We use the same DHP checkpoint as the upstream hierarchy parser in HiKEY and report F1 and STEDS on HRDoc-S, HRDoc-H, and DocHieNet.

LVLM-based multimodal DHP. Unlike many prior DHP-style components, our DHP parser is *LVLM-based multimodal*: it uses text, table crops, and figure crops as first-class inputs for hierarchy recovery. Text-only parsers cannot use visual cues, such as a table whose structure is clear only in the rendered layout or a caption spread across columns. This distinction matters for **HiKEY** because downstream retrieval treats tables and figures as first-class nodes, so their hierarchy must be recovered from multimodal signals.

Model and outputs. DHP is implemented as a structure-aware parser over page-level layout blocks. It predicts parent–child relations to reconstruct $\mathcal{T}(d)$ and identifies block types and span boundaries needed for indexing. All retrieval experiments use the same fixed DHP checkpoint. DHP is not tuned on ODQA benchmarks, so downstream gains are attributable to the retrieval framework rather than task-specific parser training.

Standalone validation of DHP reliability. A natural concern is whether the recovered hierarchy is accurate enough for routing and packing. We therefore report component-level validation for the same DHP checkpoint used in HiKEY, adapted from M3DocDep (Shin et al., 2026). Appendix Table 11 reports F1 and STEDS (Zhong et al., 2020) on HRDoc-S, HRDoc-H (Ma et al., 2023), and DocHieNet (Xing et al., 2024a). These results characterize the upstream parser; all ODQA retrieval and QA results are evaluated within the full HiKEY pipeline.

Downstream sensitivity. We also measure how much hierarchy signals matter for retrieval. The ablation study (Table 6) shows that removing hierarchy-aware indexing or routing substantially

lowers retrieval quality compared with field-separated hierarchy indexing and coarse-to-fine routing. This indicates that HiKEY's gains do not come only from larger contexts or stronger encoders; they depend on structural signals supplied by DHP.

Representative DHP failure cases and downstream impact. We identify two main residual failure modes on HRDoc-S/H and DocHieNet and trace how they affect HiKEY. (**F-H**) *Missing or misclassified headers.* If a header is detected as a paragraph, or a paragraph as a header, the affected unit may inherit a too-broad Governing Header, sometimes only the document Title. This weakens Stage-1 routing because the `Doc_card` loses a discriminative section string, and it can merge Sec_cards with neighbors, lowering Stage-2 precision. In our inspection, documents with frequent header drops tended to have lower Recall@1 even when their body-text retrieval quality was similar. (**F-N**) *Incorrect nesting.* If a subsection is attached to the wrong ancestor, the section path remains plausible but has the wrong scope. This has limited impact on coarse routing, but it can mislead the Reader when the query depends on scope, because Phase 1 packing sends the Anchor Unit with the wrong path. Ancestry-aware packing partially mitigates this by always exposing the full candidate path to the Reader. Both failure modes are reflected in the parser metrics: STEDS captures nesting errors, while F1 captures header-type errors. This is why Table 11 reports both metrics rather than a single aggregate.

DHP is an offline, one-time indexing cost. DHP runs once per document during offline indexing, not at query time. After $\mathcal{T}(d)$ and section paths are computed, they are stored in the index and reused for all future queries. Thus, deployed query-time latency is determined by Stage-1 routing, Stage-2 scoring, and Reader inference; none of these steps re-run DHP. Appendix I reports the offline cost separately from query-time latency in Table 13. In realistic deployments, the per-document DHP cost is amortized over many queries.

Discussion. DHP is an upstream module and may fail on severely degraded scans or unusual layouts. However, HiKEY is designed to remain robust by combining hierarchy-based routing, multimodal fine retrieval, and budgeted evidence packing. Table 11 bounds average parser accuracy, Table 6 isolates the contribution of hierarchy in the full pipeline, and the offline-cost analysis above addresses deployment efficiency. Together, these analyses clarify both the reliability and the practical cost of hierarchy parsing in HiKEY.

D.2 Section-path Extraction

Governing Header. For each evidence unit (block) c, we traverse upward in the recovered tree $\mathcal{T}(d)$ and choose the nearest `Title` or `Section Header` ancestor as its *governing header*.

Section Path. Using the header sequence from the root to the governing header, we define:

$$\texttt{section\_path}(c) = \texttt{Title} > \texttt{Sec} > \texttt{Subsec} > \ldots$$

This path is stored as structural metadata and used by both the hierarchy-field index and graph traversal.

D.3 `Doc_card` Construction

Stage-1 routing uses a lightweight summary representation, `Doc_card`(d), which provides global topic cues while avoiding noisy full-body text.

Deterministic Construction Rules. To implement the definition in Sec. 3 efficiently, we build `Doc_card` by concatenating: (i) the detected `Title` text, selected by DP confidence and position; (ii) high-level section headers in reading order, usually depth 1–2; and (iii) ToC entries when a Table of Contents page is detected. For indexing stability, the final string is truncated to a validation-tuned maximum length.

Design Intent. This design keeps the Stage-1 index compact while preserving strong global topic signals. It also reduces sensitivity to noisy OCR and irrelevant local details in body text.

D.4 No Explicit Cross-Block Link Modeling in HiKEY

HiKEY does not predict or construct explicit cross-block links beyond the DHP hierarchy tree. DHP is used only to recover the parent–child hierarchy $\mathcal{T}(d)$ and to provide ancestry context, such as section paths and governing headers. All cross-unit association needed for ODQA is handled deterministically during packing by the hybrid policy described below.

D.5 Semantic Associate Mining for Hybrid Packing

For each Stage-2 anchor unit c_i, we mine Semantic Associates by ranking other units in the same document with a similarity function $Sim(c_i, \cdot)$ computed from precomputed dense embeddings. Text units use text embeddings, while table and figure crops use visual embeddings. During packing, high-similarity visual units are added as Semantic Associates if they fit the token budget, and their DHP ancestry is attached so that each unit remains interpretable in its original section context.

E Ancestry-aware Evidence Subgraph Assembly

This section describes the ancestry-aware packing policy used to build the final multimodal context under a strict token budget B_{tok}.

E.1 Serialization Format

To help the Reader perceive document structure, each selected evidence unit is serialized with its structural context:

- Ancestry Context: the document Title and governing header chain from the DHP tree, which locates the unit (e.g., `# 2. Methods > ## 2.1. Model`).
- Unit Metadata: unique identifiers, unit type (Text/Table/Figure), and source page number.
- Content: the unit's textual content, such as OCR text, a linearized table, or a caption.
- Visual Crop: for `Table` and `Figure` units, the corresponding image crop is inserted when the LVLM budget allows.

E.2 Ancestry-aware Packing Algorithm

Unlike naive greedy packing, our method prioritizes the structural integrity of evidence. It first adds high-scoring Stage-2 units and their ancestry nodes, then expands with (i) Sibling Units under the same parent section and (ii) Semantic Associates selected by embedding similarity. This produces a coherent subgraph within the token budget.

What does the "subgraph" contain? Each evidence subgraph $\mathcal{S}$ contains nodes from the DHP tree $\mathcal{T}(d)$ and their ancestry chains. The nodes have three roles: (1) one or more *Anchor Units*, i.e., the highest-ranked Stage-2 units for the query; (2) the *Governing Headers* of each included unit, i.e., the chain of section titles from the document root; and (3) optional *Sibling Units* under the same parent section as an Anchor (Phase 2) and *Semantic Associates* retrieved from elsewhere in the document by embedding similarity (Phase 3). The subgraph is *not* a learned cross-block link graph. Its edges are exactly the parent–child tree edges of $\mathcal{T}(d)$ restricted to included nodes, with no predicted cross-unit links (Appendix D.4).

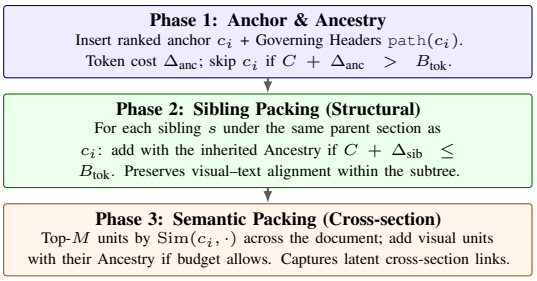

Figure 2: Schematic of the three packing phases in Algorithm 1. Each phase is budget-gated, and the loop iterates over the ranked Stage-2 anchors.

Schematic of the packing phases. Figure 2 shows the three phases of Algorithm 1 for one Stage-2 anchor. Phase 1 inserts the Anchor Unit and its Governing Headers. Phase 2 adds Sibling Units from the same parent subtree. Phase 3 adds Semantic Associates from elsewhere in the document based on embedding similarity. Each phase is constrained by the remaining budget $B_{\text{tok}} - C$; any unit that would exceed the budget is skipped, and the loop moves to the next ranked anchor.

Compact running example. We apply the three phases to the Apollo 11 query from Appendix G: *"Who were the crew members of Apollo 11, and when did they land on the Moon?"* Assume the top-ranked Stage-2 anchor is the crew paragraph c_1 (Text) under `3.1 Prime crew`.

- *Phase 1 (Anchor + Ancestry)* adds c_1 with its Governing Headers, `Apollo 11 > 3 Mission personnel > 3.1 Prime crew`, using about 180 tokens under the Qwen2.5-VL tokenizer.
- *Phase 2 (Sibling expansion)* adds the crew table c_2 and crew portrait figure c_3, both under `3.1 Prime crew`; each costs about 350–450 tokens including vision-encoder patches.

Algorithm 1 An ancestry-aware hybrid packing algorithm that assembles a coherent evidence subgraph using Sibling and Semantic strategies.

Require: Stage-2 ranked anchors $\{(c_i, s_i)\}_{i=1}^{N}$, DHP tree $\mathcal{T}(d)$, semantic similarity function $Sim(\cdot,\cdot)$, token budget B_{tok}
Ensure: Evidence subgraph $\mathcal{S}$

$\mathcal{S} \leftarrow \emptyset$, $C \leftarrow 0$ ▷ C: current accumulated token count
for $i = 1$ to N **do**
  Phase 1: Anchor & Ancestry
  $\Delta_{\text{anc}} \leftarrow$ tokens for unit c_i + Ancestry Context
  if $C + \Delta_{\text{anc}} > B_{\text{tok}}$ **then** continue
  end if
  Add c_i and Ancestry to $\mathcal{S}$; $C \leftarrow C + \Delta_{\text{anc}}$
  Phase 2: Sibling Packing (Structural)
  $Siblings \leftarrow$ {units sharing same parent section with c_i}
  for $s \in Siblings$ **do**
    $\Delta_{\text{sib}} \leftarrow$ tokens for s ▷ Sibling s inherits the same Ancestry Context as c_i
    if $C + \Delta_{\text{sib}} \leq B_{\text{tok}}$ and $s \notin \mathcal{S}$ **then**
      Add s (with inherited Ancestry) to $\mathcal{S}$; $C \leftarrow C + \Delta_{\text{sib}}$
    end if
  end for
  Phase 3: Semantic Packing (Cross-Section)
  $Candidates \leftarrow$ Top-M units by $Sim(c_i, \cdot)$ from Doc
  for $m \in Candidates$ **do**
    if m is visual unit and $m \notin \mathcal{S}$ **then**
      $\Delta_{\text{sem}} \leftarrow$ tokens for m + Ancestry
      if $C + \Delta_{\text{sem}} \leq B_{\text{tok}}$ **then**
        Add m and Ancestry to $\mathcal{S}$; $C \leftarrow C + \Delta_{\text{sem}}$
      end if
    end if
  end for
end for
return $\mathcal{S}$

- *Phase 3 (Semantic expansion)* finds that the landing-time table under `5.1 Landing` is highly similar to c_1 because the query also asks about "landing." It is added as a Semantic Associate, together with its own Governing Headers, `5 Mission events > 5.1 Landing`.

The final subgraph is $\mathcal{S} = \{c_1, c_2, c_3, c_{\text{landing}}\}$ plus two ancestry chains, and it is serialized into the Reader prompt as shown in Appendix G. Because each unit is tagged with its own ancestry, the cross-section unit from Phase 3 remains clearly scoped under `5.1 Landing`, not `3.1 Prime crew`.

Implementation Note. Token counts are computed with the Reader tokenizer (Qwen2.5-VL). Image crops receive a fixed token cost based on the vision encoder's patch tokens, and the number of images is capped to satisfy the LVLM's maximum input-resolution constraints.

F LVLM and RAG Settings

This section gives the retriever, reader, and hyperparameter settings needed for reproducibility. All experiments are implemented with PyTorch and HuggingFace Transformers.

Setting	Configuration
Infrastructure	NVIDIA H100 (80GB) × 4
Index Unit	Corpus-level Joint Indexing
Sparse Retriever	BM25 ($k_1 = 1.5, b = 0.75$)
Dense Retriever (Text)	`gte-Qwen2-7b-instruct`
Dense Retriever (Visual)	`google/siglip-large-patch16-384`
Strategy	Stage-1: Hierarchical Routing (BM25 + Dense)
	Stage-2: Fine-grained Multimodal Fusion
Reader Model	`Qwen2.5-VL-7B-Instruct`
Context Limit	16,384 tokens
Decoding	Greedy decoding (Temperature=0.0)
Evaluation Metrics	Retrieval: Recall@K, MRR@K, Hit@K, All@K
	QA: EM, ANLS, ROUGE-L, METEOR

Table 12: Experimental configuration summary for retrieval and QA. We list infrastructure, sparse/dense retrievers (text and visual), the two-stage strategy (hierarchical routing + fine-grained fusion), the LVLM reader configuration, and evaluation metrics.

F.1 Retrieval Configuration

Sparse Retrieval. We use standard BM25 settings from production search libraries such as Pyserini or Elasticsearch. After stopword removal, tokenized terms, e.g., from a morphological analyzer, are used for both Stage-1 routing fields and Stage-2 text units.

Dense Retrieval. We use gte-Qwen2-7b-instruct (Li et al., 2023) for text embeddings, with an 8192-token window for long contexts. For visual evidence units, we crop tables and figures and encode them with `SigLIP` (Zhai et al., 2023) (`google/siglip-large-patch16-384`) into 1024-dimensional embeddings. Signals are combined by late fusion with weights tuned on the validation set.

F.2 Reader Configuration

Model & Decoding. We use Qwen2.5-VL-7B-Instruct as the Reader. For reproducibility, we set

Temperature to 0.0 (greedy decoding) and cap generation at 256 tokens.

Input Context. The retrieved ancestry-aware evidence subgraph is serialized into a text prompt. The total input length is capped at 16K tokens to reflect realistic H100 memory constraints.

F.3 Prompt Template

Figure 3 shows the QA prompt template used in our experiments. The model is instructed to answer only from the provided `[Context]` (the assembled subgraph), reducing hallucination risk.

QA Prompt Template

[System Instruction]
You are an AI assistant that answers questions by analyzing the provided documents.
Write an accurate answer to `[Question]` using only the `[Context]` given below.
- Do not answer using information that is not present in the context.
- When referring to tables or figures, explicitly mention their IDs (e.g., Figure 3).
- If you cannot be confident, output "I do not have enough information to answer."

[Question]
`<QUESTION_TEXT>`

[Context]
`<SERIALIZED_EVIDENCE_SUBGRAPH>`

[Answer]

Figure 3: LVLM QA prompt template. The [Context] slot is filled with the serialized ancestry-aware evidence subgraph assembled under the token budget.

G Qualitative Case Study and Analysis

This section shows how **HiKEY** processes Wikipedia-style multimodal documents. We walk through the pipeline in Fig. 1 on an *"Apollo 11"* document and visualize the final evidence subgraph given to the LLM Reader.

G.1 System Walkthrough

For the query *"Who were the crew members of Apollo 11, and when did they land on the Moon?"*, **HiKEY** produces an answer through the following steps.

Inference Walkthrough: Apollo 11 Mission Query

(a) Document Parsing & OCR.
We parse `Apollo_11.pdf` and extract layout blocks.
- `B_h`: Headings "3. Mission personnel" and "5.1 Landing"
- `B_p`: Text "The prime crew selected for Apollo 11 consisted of..."
- `B_t`: Table "Position | Astronaut" (Commander: Neil Armstrong...)
- `B_f`: Figure (official crew portrait) plus crop

(b) Hierarchy & Graph Construction.
DHP reconstructs the ToC-like structure and assigns section paths.
- Path: `Apollo 11` → `3 Mission personnel` → `3.1 Prime crew`
- Units: c_1(Text), c_2(Table), c_3(Figure)
- Edges: Tree(`Sec 3.1` → c_1, c_2, c_3)

(c) Stage-1: Hierarchical Routing.
The query terms "crew" and "landing" are matched against the hierarchy field (`Doc_card`) to narrow the scope.
- Output: Target Doc={`Apollo_11.pdf`}, Anchors={`Sec 3.1`, `Sec 5.1`}

(d) Stage-2: Fine Retrieval & Assembly.
We retrieve units under the anchors and pack the text c_1, the crew table c_2, and the landing-time table under the token budget B_{tok}. The crew table is added as a Sibling Unit, while the landing-time table is added as a Semantic Associate.

G.2 Serialized Input Visualization

The selected evidence units are serialized and passed to the LVLM. The example below shows how the text explanation, crew table, and linked crew photo are combined coherently.

Example of Serialized Evidence Subgraph (Input Prompt)

```
# Global Context
[DOC_META]
ID: Apollo_11.pdf | Title: Apollo 11 -
Wikipedia

---

# Hierarchy Anchor (Routing Result)
[SCOPE] 3. Mission personnel > 3.1. Prime
crew

[UNIT id=42 | Type=Text]
Path: 3. Mission personnel > 3.1. Prime
crew
Content: The prime crew for Apollo 11
consisted of Commander Neil Armstrong, Command
Module Pilot Michael Collins, and Lunar Module
Pilot Edwin "Buzz" Aldrin. Their positions
are detailed in Table 1.

[UNIT id=43 | Type=Table]
Path: 3. Mission personnel > 3.1. Prime
crew
Visual: <|image_token_1|> (Table Crop)
```

[IMAGE PLACEHOLDER]
(Cropped image of Table 1: Prime Crew Members)

⇓ Sibling Evidence (Same Section) ⇓

```
[UNIT id=45 | Type=Figure]
Path: 3. Mission personnel > 3.1. Prime
crew
Caption: Figure 2. The official crew
portrait of Apollo 11.
```

Visual: <|image_token_2|> (Photo Crop)

H Qualitative Failure Cases of Baseline Families

We summarize three representative failure modes observed when inspecting baseline outputs on FRAMES and M3DocVQA. These examples make the structural limits in Table 7 concrete, but they are not exhaustive.

(F1) Chunk-RAG: routing locked onto a lexically similar but topically wrong document. For composite FRAMES queries with shared named entities across unrelated articles, such as multiple Wikipedia pages mentioning "Apollo," flat chunk retrievers often rank chunks from a nearby but wrong document at the top. The similarity signal is dominated by local token overlap. Once the wrong document is selected, later chunks reinforce the error, which helps explain why chunk-RAG's FRAMES *All* score is much lower than its Hit score. **HiKEY** avoids this by routing at the `Doc_card` level, where section paths separate topically different documents even when body text looks similar.

(F2) Page-level multimodal RAG: correct page retrieved, answer drowned by page-wide noise. Page-embedding methods may retrieve the correct page but still fail in end-to-end QA. A fixed page unit injects unrelated text and visual noise into the 16K budget, which can push the answer span below the model's attention priority. This is consistent with the gap between retrieval Hit@K and end-to-end EM in Tables 1 and 2. **HiKEY** instead packs a block-level Anchor Unit with only the ancestry needed for interpretation, leaving budget for sibling and semantic expansion.

(F3) Text-only GraphRAG: multi-hop succeeds on text but breaks on table/figure evidence. RAPTOR and HopRAG connect text chunks, but they do not expose tables and figures as first-class retrieval targets. For M3DocVQA questions whose answer appears in a table row or figure caption, these systems may retrieve nearby text that *mentions* the table but not the table content itself. **HiKEY** treats each table or figure crop as a retrieval unit with Upper Context, so the answer-bearing unit can be ranked and packed directly. See Tab. 3 for the Table/Image breakdown.

Stage	Latency (s)			Complexity
	5 pages	10 pages	20 pages	
Visual Analysis (High-Res)				
Layout Detection	5.2	10.5	21.2	$O(P)$
OCR (Dense Text)	24.5	48.2	98.4	$O(P \times T)$
Visual Embedding	12.5	25.8	52.6	$O(N_{\text{img}})$
Structure & Graph Construction				
DHP Tree Parse	0.5	1.0	2.1	$O(N_{\text{blk}})$
Graph Build	0.2	0.4	0.9	$O(N + E)$
Total Indexing Time	42.8	85.8	175.2	Linear w.r.t Pages

Table 13: Offline indexing runtime and scalability as a function of document length. We report stage-wise latency (layout detection, OCR, visual embeddings, DHP parsing, and graph building) for 5/10/20-page documents and provide complexity terms indicating the main cost drivers.

I Runtime and Scalability Analysis

We analyze the computational cost of **HiKEY** in a realistic industrial setting. Measurements use a single NVIDIA H100 (80GB) GPU, 144 DPI rendering, and high-precision OCR. Peak GPU memory during offline indexing is about 27 GB.

I.1 Offline Indexing Cost

Table 13 breaks down runtime by document length, including high-resolution layout detection, OCR, visual embedding, DHP tree parsing, and graph construction. The full offline pipeline costs about 8–9 s/page on H100 in this high-precision setting. The core DHP tree parser is lightweight, about 0.1 s/page, so most cost comes from OCR and visual embedding at index time rather than query time.

Findings and Cost Justification. More than 90% of indexing time is spent on visual analysis, including layout detection, OCR, and embeddings. This cost is expected when preserving complex industrial documents at high resolution. The core **HiKEY** logic—tree parsing and graph construction—is lightweight ($< 2\%$ of total time) and does not create a bottleneck. Thus, **HiKEY** organizes expensive vision-derived signals into a structured index with little additional overhead.