LAD-RAG: Layout-aware Dynamic RAG for Visually-Rich Document Understanding

Zhivar Sourati[1,2], Zheng Wang[2], Marianne Menglin Liu[2], Yazhe Hu[2], Mengqing Guo[2]

Sujeeth Bharadwaj[2], Kyu Han[2], Tao Sheng[2], Sujith Ravi[2], Morteza Dehghani[1] and Dan Roth[2]
[1]University of Southern California
[2]Oracle AI

Abstract

Question answering over visually rich documents (VRDs) requires reasoning not only over isolated content but also over documents' structural organization and cross-page dependencies. However, conventional retrieval-augmented generation (RAG) methods encode content in isolated chunks during ingestion, losing structural and cross-page dependencies, and retrieve a fixed number of pages at inference, regardless of the specific demands of the question or context. This often results in incomplete evidence retrieval and degraded answer quality for multi-page reasoning tasks. To address these limitations, we propose LAD-RAG, a novel Layout-Aware Dynamic RAG framework. During ingestion, LAD-RAG constructs a symbolic document graph that captures layout structure and cross-page dependencies, adding it alongside standard neural embeddings to yield a more holistic representation of the document. During inference, an LLM agent dynamically interacts with the neural and symbolic indices to adaptively retrieve the necessary evidence based on the query. Experiments on MMLongBench-Doc, LongDocURL, DUDE, and MP-DocVQA demonstrate that LAD-RAG improves retrieval, achieving over 90% perfect recall on average without any top-k tuning, and outperforming baseline retrievers by up to 20% in recall at comparable noise levels, yielding higher QA accuracy with minimal latency.

1 Introduction

Performing NLP tasks such as question answering, summarization, and data extraction over visually rich documents (VRDs; Xu et al., 2020) requires processing text, figures, charts, and reasoning over layout (e.g., document structure, reading order, and visual grouping; Masry et al., 2022; Wu et al., 2024; Xu et al., 2023; Kahou et al., 2017; Luo et al., 2024). Modern multimodal models (e.g., GPT4o; Hurst et al., 2024, InternVL; Chen et al., 2024,
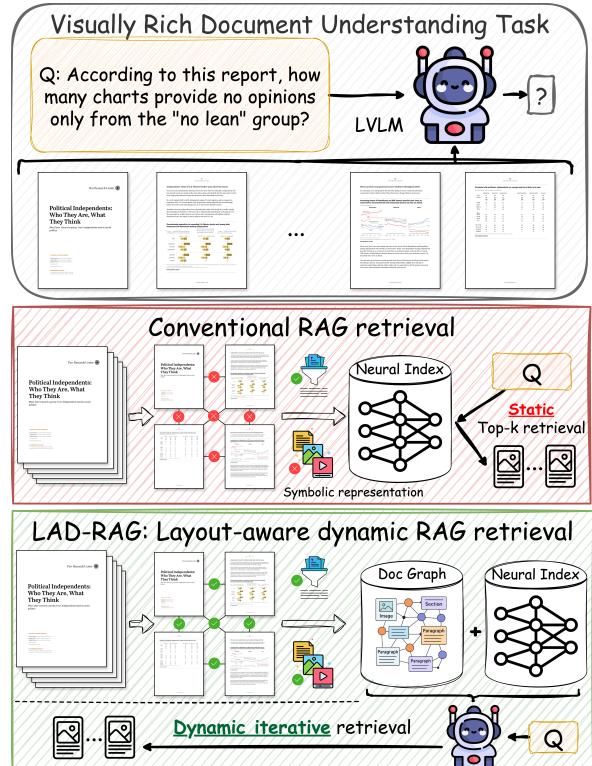

Figure 1: LAD-RAG addresses three key limitations of conventional RAGs in VRDs by (1) introducing a symbolic document graph to capture layout and cross-page structure, (2) integrating symbolic and neural indices to preserve structural and semantic signals, and (3) leveraging an LLM agent for dynamic, query-adaptive retrieval beyond static top-k methods.

Qwen2.5VL; Bai et al., 2025) are capable of processing such inputs (Wang et al., 2025a), but their effectiveness is limited when documents exceed the model's context window (Yin et al., 2024). Plus, performance often degrades as input grows longer, even when the entire document fits within the context (Zhou et al., 2024; Wang et al., 2024a), as relevant signals become diluted by noise, leading to incomplete or incorrect answers (Sharma et al., 2024; Zhang et al., 2023).

To overcome these issues, retrieval-augmented

generation (RAG; Lewis et al., 2020; Shi et al., 2023) frameworks index document chunks during ingestion and retrieve a subset of relevant chunks at inference time to help generate and ground the answer in a given document (Gao et al., 2023; Chen et al., 2022). However, as shown in Figure 1, conventional RAG approaches rely mostly on dense text and image encoders (Khattab and Zaharia, 2020; Faysse et al., 2024; Karpukhin et al., 2020), and treat document segments as a linear sequence of isolated units (Duarte et al., 2024; Płonka et al., 2025; Han et al., 2024), neglecting document structure (Wang et al., 2025b) and inter-page relationships. This results in three major limitations:

1. **Loss of layout and structural context**. Ignoring layout-driven hierarchies and cross-page continuity can lead to incomplete evidence retrieval. For example, in response to *"How many businesses are shown as examples of transformation by big data?"*, conventional retrievers may only return the title slide, missing subsequent pages listing the examples. This happens as the examples are connected with the title slide through the document's structure and layout.

2. **Over-reliance on embeddings**. Retrieval struggles with queries depending on symbolic or structural cues (e.g., references to charts, page numbers, or table sources). Consider the query *"How many charts and tables in this report are sourced from Annual totals of Pew Research Center survey data?"* Answering this requires aggregating non-contiguous, yet structurally related, figures and captions that are not explicitly captured in semantic embeddings.

3. **Static top-k retrieval**. Retrieval depth is agnostic to question or document complexity, often leading to retrieving too much or too little evidence. Consider the differing scope needed for various questions: *How many organizations are introduced in detail?"* need only three pages, but *How many distinct Netherlands location images are used as examples?"* require twelve.

In short, existing RAG pipelines lack a holistic document representation at ingestion time, which hinders the retrieval of a complete set of evidence, particularly when relevant content is structurally dispersed across the document (Ma et al., 2024; Deng et al., 2024). To address this, we introduce

LAD-RAG, a layout-aware dynamic RAG. As illustrated in Figure 1, LAD-RAG enhances conventional neural indexing with a symbolic document graph built at ingestion. Nodes in this graph encode explicit symbolic elements (e.g., headers, figures, tables), while edges capture structural and layout-based relationships (e.g., section boundaries, figure–caption links, and cross-page dependencies). This design supports both fine-grained retrieval over individual nodes and higher-level retrieval over structurally grouped elements (e.g., all components of a section), enabling multiple complementary retrieval pathways.

Given a query at inference time, to effectively leverage the indexed information, a language model agent accesses both the neural index and the document graph to determine an appropriate retrieval strategy: neural, graph-based, or hybrid, and iteratively interacts with both indices to retrieve a complete set of evidence. Critically, because the document graph encodes both local and global structural relationships, including layout-based neighborhoods and higher-order patterns such as community partitions, LAD-RAG supports contextualization of retrieved nodes into coherent and complete groups of nodes. This supports the extraction of complete and well-structured evidence sets, in contrast to the partial and fragmented retrievals typical of traditional RAG systems.

We evaluate LAD-RAG on four challenging VRD benchmarks, MMLongBench-Doc, LongDocURL, MP-DocVQA, and DUDE, with diverse layouts (slides, reports, papers) and questions requiring evidence scattered across multiple pages. LAD-RAG consistently improves both retrieval (achieving over 90% perfect recall on average without any top-k tuning, and outperforming baseline retrievers by up to 20% in recall at comparable noise levels) and QA accuracy, approaching the performance with ground-truth evidence, while introducing minimal inference latency.

2 LAD-RAG framework

Given a document (d) that is the target of a question (q), the goal of our LAD-RAG framework is to construct a *holistic, contextualized* understanding of d's content at ingestion time to support more complete and accurate retrieval at inference time. Achieving this requires organizing the framework into two corresponding phases, illustrated in Figure 2: (i) preparing and storing rich document in-

formation both neurally and symbolically during **ingestion** (Section 2.1), and (ii) using that information strategically during **inference** (Section 2.2).

2.1 Ingestion

At ingestion time, we aim to comprehensively parse the document d and store its content in both neural and symbolic forms. Inspired by how human readers sequentially build mental models by reading page by page while tracking cross-references and higher-order themes (Saux et al., 2021), we simulate a similar process, as demonstrated in Figure 2.

To support flexible understanding across diverse document layouts, we use GPT-4o (Hurst et al., 2024), a powerful large vision–language model (LVLM), to process each page in sequence (see Appendix G for details). As the LVLM parses each page, it extracts all visible elements and generates self-contained descriptions for each (which will later form the **nodes** of the document graph).

Beyond page-level processing, we maintain a running **memory** (M), akin to a human reader's ongoing understanding of a document (Saux et al., 2021). This memory accumulates key high-level information, e.g., section structure, entity mentions, and thematic progressions, across pages. As each new page is processed, we connect its elements with relevant parts of M to build inter-page relationships, as the **edges** in the document graph.

Once the model has completed a full pass over the document, we construct a full **document graph** (G) containing both intra- and inter-page structure. This graph is stored in two complementary forms: the full symbolic document graph object and a neural index over its nodes, each enabling different modes of downstream retrieval.

Document graph nodes. Each node corresponds to a localized element on a page, such as a paragraph, figure, table, section title, or footnote. For every element, we extract Layout position on the page, Element type (e.g., figure, paragraph, section header, etc.), Displayed content, including extractive text or semantic, captions, Self-contained summary that enables standalone interpretation, and Visual attributes (e.g., font, color, size)

This node-level representation ensures that every element can be retrieved and interpreted in isolation (Bhattacharyya et al., 2025), if necessary, while also enabling structured indexing.

Document graph edges. Edges connect nodes based on symbolic, layout, or semantic relation-

ships, including: Reference relationships, e.g., a paragraph referring to a figure, or a footnote referring to a section, and Layout/structural relationships, e.g., elements belonging to the same section or representing cross-page continuations.

Constructing these edges requires a higher-level understanding of the document's structure. This is enabled by the running memory (M), which tracks the evolving context, including the current section hierarchy, the active entities under discussion, and unresolved references (e.g., placeholders for elements explained later). These contextual signals help disambiguate relationships that are not evident from the current page alone.

Neural–symbolic indexing. The final output of ingestion is stored in two complementary representations: 1. Symbolic (graph) index (G): A document graph with structured node/edge properties (e.g., content type, location, visual attributes) and explicit local and global relationships for graph-based retrieval, capturing a set of nodes part of a community captured by the graph structure. 2. Neural index (E): A vector-based index over the self-contained summaries of all nodes, enabling semantic similarity search.

This dual representation preserves both the semantic richness of neural models and the explicit structure captured in the document's layout, enabling retrieval mechanisms that would otherwise be inaccessible through embedding-based approaches alone, which can summarize and abstract away from the input content, making nuances and details explicitly inaccessible.

2.2 Inference

At inference time, given a question q, its symbolic graph G, and its neural index E, the goal is to retrieve a complete and contextually appropriate set of evidence from G and/or E. This process is both query-dependent (some questions can be answered via embedding similarity, while others require symbolic reasoning over layout or structure) and interactive, as the scope and difficulty of the query may require varying amounts of evidence.

To accommodate these needs, LAD-RAG employs an LLM agent (GPT-4o, Hurst et al., 2024) that iteratively interacts with both indices to retrieve the full set of evidence required to answer the question (see Figure 2).

The LLM agent is equipped with tool interfaces with explicit function signatures to access and oper-
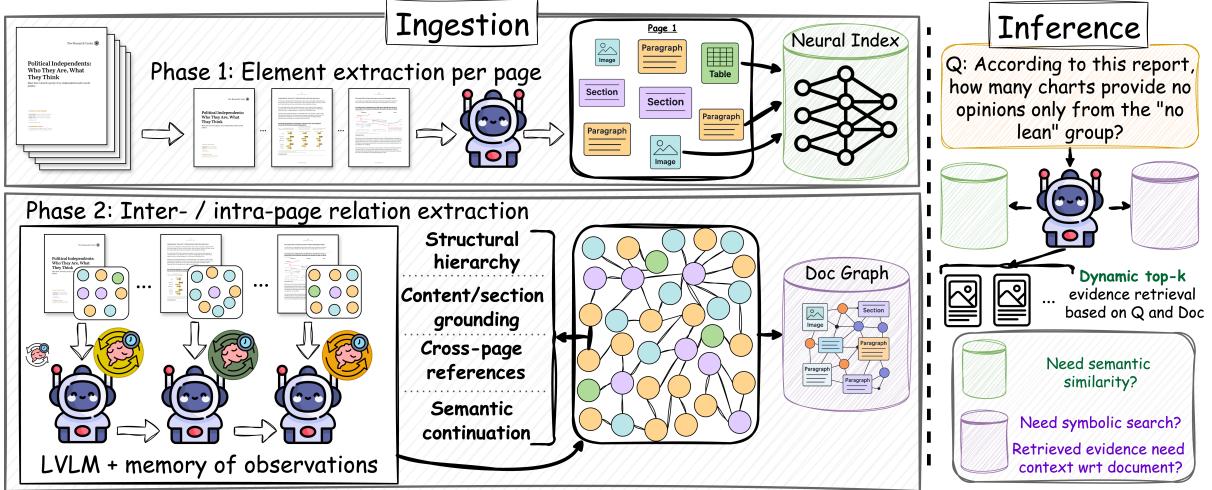
Figure 2: LAD-RAG framework: During ingestion, an LVLM extracts elements from each page and encodes them into an index, while also constructing a document graph that captures inter- and intra-page relationships for symbolic search. During inference, an agent interprets the question and iteratively interacts with both the neural index and the document graph to retrieve relevant evidence, enabling question-aware, structure-aware retrieval.

ate over the indices. Given the query, the agent generates a high-level plan (e.g., selecting between semantic, symbolic, or hybrid retrieval modes), issues tool calls accordingly, and iteratively refines the evidence set through a conversational loop. This loop terminates when one of the following conditions is met: (i) nearing the model's context window, (ii) reaching a maximum number of steps, or (iii) the agent determines that sufficient evidence has been gathered. See Appendix G for details.

The following tools are exposed to the agent: 1. **NeuroSemanticSearch(query):** Retrieves evidence based on embedding similarity from the neural index, using a query composed by the agent based on the given question q. 2. **SymbolicGraph-Query(query_statement):** Performs structured queries over the symbolic document graph (e.g., filtering by element type, section, or page). The agent is instructed on the graph representation and must generate query statements to interact with the document graph object and extract relevant nodes based on their properties or structural position. 3. **Contextualize(node):** Expands a given node into a broader evidence set based on its structural proximity within the graph. This expansion leverages both local neighborhoods and higher-order relationships, using Louvain community detection (Blondel et al., 2008) to surface coherent clusters of contextually related nodes. Additional implementation details are provided in Appendix G.

Together, these tools enable the system to flexibly retrieve evidence tailored to the specific query,

document, and reasoning complexity, going beyond fixed top-k retrieval to support fully contextualized and adaptive evidence selection.

3 Experimental Setup

3.1 Datasets

To evaluate LAD-RAG, we conduct experiments on four diverse benchmarks for visually rich document understanding: MMLongBench-Doc (Ma et al., 2024), LongDocURL (Deng et al., 2024), DUDE (Deng et al., 2024), and MP-DocVQA (Tito et al., 2023). These datasets cover a wide range of domains and question types, including those grounded in high-level document content and those dependent on more localized visual elements such as figures, references, sections, and tables. They also vary in the number of pages required to answer questions, making them well-suited for evaluating retrieval completeness as well as the downstream question answering performance of LAD-RAG. See Appendix B for details.

3.2 Baselines

Retrieval. We compare LAD-RAG's evidence retrieval performance with both text-based and image-based retrievers. The text-based baselines include E5-large-v2 (Wang et al., 2022), BGE-large-en (Xiao et al., 2024), and BM25 (Lù, 2024), while the image-based baseline is ColPali (Faysse et al., 2024). These baselines follow the standard RAG setup, operating over summaries of all extracted page elements, retrieving the top-k elements based

on similarity to the query, at inference time. We evaluate performance across k values up to the point of perfect recall.

Question answering. To assess the downstream QA performance of our system, we pair the retrieved evidence with four LVLMs: Phi-3.5-Vision-4B (Abdin et al., 2024), Pixtral-12B-2409 (Agrawal et al., 2024), InternVL2-8B (Chen et al., 2024), and GPT-4o (Hurst et al., 2024).[1] For each model, we use deterministic greedy decoding for answer generation and compare QA accuracy under several retrieval settings:

- Using evidence retrieved by LAD-RAG.

- Using the best-performing baseline retriever at fixed retrieval sizes: $k = 5$ and $k = 10$.

- Using the same best-performing baseline retriever, matching the number of retrieved items with LAD-RAG (to control for retrieval budget).

- Using ground-truth evidence pages (oracle retrieval) as an upper bound.

To illustrate the value of retrieval for QA in VRD, we also compare against models that receive the full document as input, i.e., mPLUG-DocOwl v1.5-8B (Hu et al., 2024), Idefics2-8B (Laurençon et al., 2024), and MiniCPM-Llama3-V2.5-8B (Yao et al., 2024), highlighting the effectiveness of retrieval-augmented approaches in reducing noise while maintaining high accuracy.

3.3 Evaluations & Metrics

Retrieval. Let a question q be associated with a document d, and let the ground-truth set of evidence pages be denoted by $P = \{p_1, p_2, \ldots, p_n\}$. Given a retriever that returns a set of pages $\hat{P}$, our goal is to assess how well $\hat{P}$ matches P, both in terms of completeness and precision. We report two key retrieval metrics:

- **Perfect Recall (PR):** A binary metric defined as:

$$ PR = \begin{cases} 1 & \text{if } P \subseteq \hat{P} \\ 0 & \text{otherwise} \end{cases} $$

It indicates whether the retriever has retrieved *all* ground-truth pages. This is crucial for multi-page questions, where missing even a single evidence page may result in incorrect answers.

- **Irrelevant Pages Ratio (IPR):** The proportion of retrieved pages that are *not* in the gold set:

$$ IPR = \frac{|\hat{P} \setminus P|}{|\hat{P}|} $$

capturing the noise the retriever introduces, with lower values indicating a more targeted retrieval.

An ideal retriever achieves high perfect recall while minimizing the irrelevant pages ratio, ensuring that all necessary evidence is retrieved without excessive inclusion of irrelevant content.

Question answering. Following the evaluation setup in MMLongBench (Ma et al., 2024) and LongDocURL (Deng et al., 2024), we use GPT-4o (Hurst et al., 2024) to extract concise answers from retrieved content and apply a rule-based comparison to assign binary correctness, based on which we report **accuracy**.

4 Results

4.1 Retrieval Effectiveness of LAD-RAG

We evaluate the retrieval performance of LAD-RAG across four challenging VRD benchmarks and baseline methods in Figure 3. Notably, LAD-RAG achieves over 90% perfect recall on average across datasets *without any top-k tuning*. Despite this, it retrieves significantly fewer irrelevant pages compared to existing baselines. At equivalent irrelevance rates, LAD-RAG outperforms other baselines by approximately 20% on MMLongBench-Doc, 15% on LongDocURL, and 10% on both DUDE and MP-DocVQA, in the perfect recall rate.

In contrast, baseline retrievers require large top-k values to match our recall rate: on average, $k = 25$ for MMLongBench-Doc, $k = 29$ for LongDocURL, $k = 10$ for DUDE, and $k = 5$ for MP-DocVQA. These findings reveal a critical mismatch between common retrieval practices (where k is typically capped at low numbers; Cho et al., 2024; Han et al., 2025) and the actual evidence volume required to answer multi-page questions. This also partially explains the low QA performance (30–40% accuracy) reported by state-of-the-art models on complex VRD benchmarks such as MMLongBench-Doc (Ma et al., 2024) and LongDocURL (Deng et al., 2024).

The advantage of LAD-RAG is even more pronounced in questions requiring evidence from multiple pages. LAD-RAG consistently retrieves more
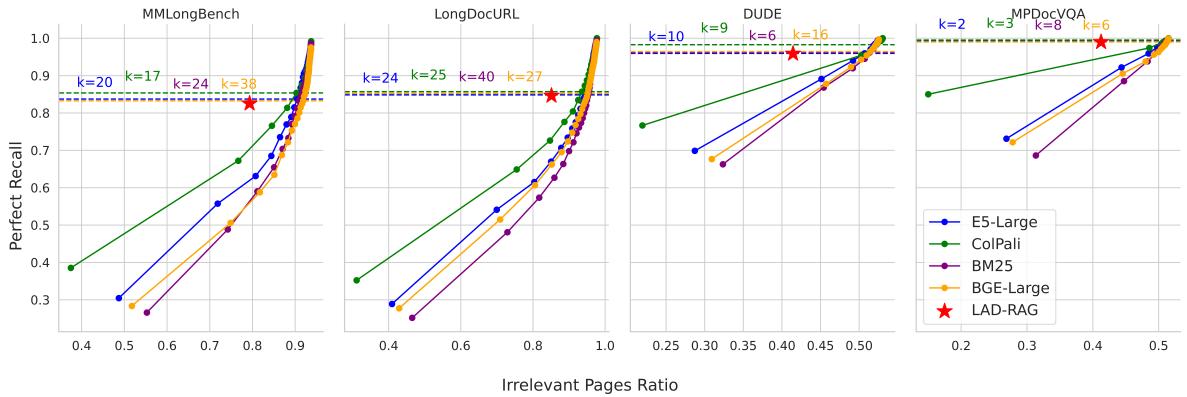
---

[1]We also tested LLaMA 3.2 Vision but filtered it out after it failed multi-image perception tasks in preliminary testing.

Figure 3: Retrieval performance of LAD-RAG compared to baseline retrievers across varying top-k settings. Baselines retrieve from $k = 1$ up to the point of perfect recall. Dotted horizontal lines indicate the number of retrieved pages each baseline requires to match the recall achieved by LAD-RAG without any top-k tuning.

complete multi-page evidence sets than all baselines (see Appendix D). This reflects its ability to capture distributed information through structured layout modeling and dynamic interaction with the symbolic and the neural indices. For a deeper understanding of these benefits, we present detailed case studies in Appendix C.

4.2 Ablation Study of Retrieval Components

To understand the contribution of individual components in LAD-RAG, we conduct an ablation study on MMLongBench and LongDocURL, the most challenging benchmarks, comparing overall retrieval performance (measured as the ratio of perfect recall to irrelevant page retrievals, with higher values indicating better recall and lower noise) across LAD-RAG variants. We compare the full LAD-RAG (with both `Contextualize` and `GraphQuery`) against versions that disable one or both components. As shown in Table 1, removing either contextualization (C) or symbolic graph querying (G) results in noticeable performance drops: at similar noise levels, recall decreases by an average of 4% without contextualization and 10% without graph querying. Notably, the variant without graph or contextualization (LAD-RAG w/o C & G), which is a retrieval agent interacting only with the neural index, still outperforms conventional baselines. These results show that while symbolic and graph-based mechanisms are essential, the interactive nature of the retriever also contributes to assembling a complete evidence set.

4.3 LAD-RAG's End-to-End QA Gains

While our core contribution is retrieval, we also observe consistent gains in downstream QA accu-

Method	MMLongBench	LongDocURL
LAD-RAG	0.979	0.895
LAD-RAG w/o C	0.957	0.819
LAD-RAG w/o G	0.856	0.809
LAD-RAG w/o C & G	0.840	0.774
ColPali	0.831	0.791
BM25	0.728	0.762
E5-Large	0.791	0.769
BGE-Large	0.743	0.704

Table 1: Retrieval performance of LAD-RAG variants, measured as the ratio of perfect recall to irrelevant page retrievals (higher is better). C and G denote `Contextualize` and `GraphQuery` operations.

racy across all models and benchmarks when using LAD-RAG. Results are shown in Table 2. Across both smaller models (e.g., InternVL2-8B, Phi-3.5-Vision) and larger models (e.g., GPT-4o-200B), LAD-RAG consistently improves QA accuracy over all retrieval baselines and the ones that access the entire document without retrieval. It outperforms both fixed top-k retrieval ($k = 5, 10$) and the topk-adjusted setting, where baselines retrieve the same number of pages as our system. This shows that our retriever not only supplies more relevant evidence but also introduces substantially less noise, directly translating to higher downstream accuracy.

Focusing on the multi-page questions that constitute the more challenging subset of examples, LAD-RAG yields consistent gains across all four benchmarks, averaging 4 and up to 18 percentage points over top-k baselines, and 3 and up to 11 points over top-k-adjusted retrieval (see Appendix E for similar trends across all question types). This indicates that LAD-RAG's improvements stem not just from retrieving more pages, but from retrieving more

| Model | Retrieval | MMLongBench-Doc | | | LongDocURL | | | DUDE | | | MP-DocVQA |
		all	single	multi	all	single	multi	all	single	multi	all
mPLUG-DocOwl 1.5-8B	All-Pages	0.069	0.074	0.064	0.031	0.039	0.024	0.150	0.188	0.116	0.150
Idefics2-8B	All-Pages	0.070	0.077	0.072	0.045	0.054	0.038	0.170	0.205	0.130	0.170
MiniCPM-Llama3-V2.5-8B	All-Pages	0.085	0.095	0.095	0.060	0.066	0.053	0.190	0.210	0.150	0.190
InternVL2-8B	Ground-Truth	0.399	0.506	0.250	0.629	0.684	0.580	0.641	0.697	0.574	0.848
	retrieving@5	0.287	0.372	0.164	0.443	0.484	0.404	0.560	0.564	0.367	0.737
	retrieving@10	0.319	0.395	0.208	0.457	0.499	0.420	0.576	0.591	0.350	0.759
	topk-adjusted	0.304	0.365	0.212	0.468	0.515	0.411	0.589	0.609	0.416	0.773
	LAD-RAG	**0.448**	**0.495**	**0.242**	**0.477**	**0.518**	**0.438**	**0.630**	**0.669**	**0.533**	**0.792**
Pixtral-12B-2409	Ground-Truth	0.498	0.537	0.343	0.634	0.668	0.603	0.602	0.636	0.467	0.839
	retrieving@5	0.389	0.395	0.200	0.430	0.458	0.401	0.514	0.512	0.417	0.663
	retrieving@10	0.345	0.327	0.178	0.427	0.457	0.400	0.529	0.560	0.417	0.655
	topk-adjusted	0.383	0.394	0.213	0.455	0.488	0.423	0.535	0.574	0.426	0.675
	LAD-RAG	**0.415**	**0.462**	**0.261**	**0.507**	**0.551**	**0.468**	**0.545**	**0.578**	**0.433**	**0.704**
Phi-3.5-Vision-4B	Ground-Truth	0.383	0.492	0.227	0.631	0.659	0.606	0.578	0.638	0.446	0.810
	retrieving@5	0.315	0.399	0.192	0.476	0.485	0.468	0.562	0.615	0.422	0.719
	retrieving@10	0.302	0.381	0.189	0.479	0.483	0.476	0.576	0.643	0.450	0.737
	topk-adjusted	0.336	0.400	0.199	0.469	0.484	0.455	0.582	0.656	0.466	0.737
	LAD-RAG	**0.391**	**0.414**	**0.202**	**0.489**	**0.493**	**0.476**	**0.596**	**0.656**	**0.467**	**0.754**
GPT-4o-200b-128	Ground-Truth	0.696	0.693	0.565	0.714	0.746	0.686	0.807	0.830	0.633	0.895
	retrieving@5	0.575	0.607	0.303	0.590	0.684	0.510	0.707	0.732	0.534	0.825
	retrieving@10	0.610	0.637	0.372	0.622	0.702	0.552	0.706	0.732	0.535	0.819
	topk-adjusted	0.593	0.629	0.409	0.652	0.709	**0.600**	0.720	**0.750**	0.541	**0.833**
	LAD-RAG	**0.625**	**0.676**	**0.450**	**0.659**	**0.724**	0.599	**0.725**	0.746	**0.545**	0.829

Table 2: Accuracy scores per model across retrieval types (topk-adjusted: evidence with the same number of retrieved pages as LAD-RAG) and top-k. Single/multi refer to questions requiring evidence from one or multiple pages, respectively. Best values per model group (excluding GT) are shown in **bold**.

focused and complete evidence.

Notably, our retrieval performance approaches that of using ground-truth evidence across all datasets, narrowing the gap to within 5–8 points, especially for challenging benchmarks like MM-LongBench and LongDocURL. This contextualizes both the upper bound of current models and the significance of improved retrieval.

Together, these results confirm that LAD-RAG improves end-to-end QA, particularly in multi-page settings where conventional methods fall short. By assembling more focused and relevant evidence, LAD-RAG boosts answer accuracy in a way that generalizes across models and datasets.

4.4 Latency Analysis

Analyzing the latency of LAD-RAG, we find that it introduces minimal overhead during inference. Graph construction is performed once, offline during ingestion, and does not affect inference-time latency. During inference, our agent-based retriever typically issues 2–5 LLM calls (Figure 4), with over 97% of these generating fewer than 100 tokens on average (Appendix F). These tokens serve as retrieval queries executed over a pre-built symbolic graph and semantic index, both incurring neg-
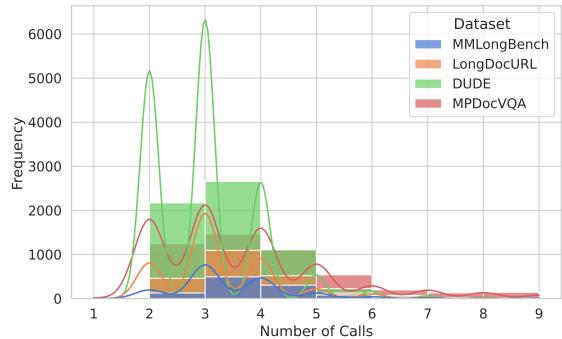

Figure 4: Distribution of the number of LLM calls per query by LAD-RAG.

ligible runtime cost. Altogether, this demonstrates that LAD-RAG achieves substantial QA gains with minimal added latency at inference time.

5 Related Work

5.1 Visually-rich Document Understanding

With the rise of LVLMs (Hurst et al., 2024; Team et al., 2023; Bai et al., 2025; Grattafiori et al., 2024; Liu et al., 2023), these models have been increasingly applied to document understanding tasks (Ma et al., 2024; Hui et al., 2024; Deng et al., 2024),

especially for documents containing rich visual content (Kahou et al., 2017; Xu et al., 2023) or spanning multiple pages (Hu et al., 2024). To reduce the reasoning burden on LLMs and improve response accuracy, RAG has become a dominant strategy, narrowing the context to only the most relevant parts of a document.

Several recent works have tackled the challenge of retrieving multi-modal, cross-page evidence in visually rich documents. To capture the multi-modal nature of these tasks, M3DocRAG (Cho et al., 2024) and MDocAgent (Han et al., 2025) combine text- and image-based retrievers with specialized reasoning agents (Khattab and Zaharia, 2020; Faysse et al., 2024). For modeling cross-page relationships by enriching the indices at ingestion time (Chen et al., 2025), MM-RAG (Gong et al., 2025) and RAPTOR (Sarthi et al., 2024) construct hierarchical aggregations of semantically related content to improve retrieval granularity. MoLoRAG (Wu et al., 2025) instead expands evidence at inference time by traversing semantically similar pages. More recently, dynamic retrieval has emerged: SimpleDoc (Jain et al., 2025) iteratively refines evidence using LLM feedback, while FRAG (Huang et al., 2025) performs a full-document pass to select relevant pages or images, addressing the limitations of fixed top-k retrieval.

While these methods have advanced multi-modal retrieval, handling of cross-page dependencies still largely relies on capturing semantically similar content across pages; ingestion pipelines depend largely on embedding-based representations, losing a symbolic view of document structure; and as a result, there is no adaptability to the specific retrieval demands of different queries. LAD-RAG bridges these gaps by introducing a symbolic document graph capturing both local content and global layout structure, paired with a query-adaptive retriever that dynamically reasons over both neural and symbolic indices to improve retrieval.

5.2 RAG Frameworks with Graph Integration

Recent RAG frameworks have enhanced retrieval through graphs. At ingestion time, several methods construct hierarchical groupings, typically based on semantic similarity or high-level layout cues like document sections, to support multi-granularity retrieval (Nguyen et al., 2024; Wang et al., 2025b; Lu et al., 2025; Sarthi et al., 2024; Edge et al., 2024). However, these structures often lack flexibility for ad hoc queries that do not align with predefined

boundaries. Other approaches incorporate graphs at inference time, using entity-based traversal (Kim et al., 2024; Niu et al., 2024; Guo et al., 2024). Iterative graph-based retrieval has also been explored (Sun et al., 2023), along with knowledge graph prompting in multi-document QA (Wang et al., 2024b; Jiang et al., 2024; Yang et al., 2025), though mainly text-only and unsuited for visually-rich documents with unpredictable structure.

LAD-RAG addresses the unique challenges of visually rich documents by constructing a general-purpose symbolic graph capturing both semantic and layout-based relationships, and not just localized semantic groups. At inference time, it performs dynamic, query-driven retrieval over this structure without costly inference-time traversal. By leveraging document structure to flexibly retrieve relevant node groups beyond rigid top-k constraints, LAD-RAG enables comprehensive evidence gathering across pages, leading to substantial gains in QA accuracy.

6 Conclusion

We introduce LAD-RAG, a layout-aware dynamic RAG framework for visually rich document understanding. Unlike conventional RAG pipelines that ingest document chunks in isolation and rely solely on neural indices, LAD-RAG constructs a symbolic document graph during ingestion to capture both local semantics and global layout-driven structure. This symbolic graph is stored alongside a neural index over document elements, enabling an LLM agent at inference time to dynamically reason over and retrieve evidence based on the specific needs of each query. Across four challenging VRD benchmarks, MMLongBench-Doc, LongDocURL, DUDE, and MP-DocVQA, LAD-RAG improves retrieval completeness, achieving over 90% perfect recall on average without any top-k tuning, and outperforming strong text- and image-based baseline retrievers by up to 20% in recall at comparable noise levels. These gains translate to higher downstream QA accuracy, approaching oracle-level performance with minimal added latency. Our results underscore the importance of reasoning over layout and cross-page structure. LAD-RAG provides a generalizable foundation for retrieval in tasks that require contextual, multimodal understanding, with broad applicability across enterprise, legal, financial, and scientific domains.

7 Limitations

LAD-RAG is designed to improve retrieval completeness and precision in RAG pipelines for visually rich document understanding. While our results show consistent gains in both retrieval quality and downstream QA accuracy, we also observe that even with near-perfect evidence, current LVLMs still exhibit limitations in fully utilizing the retrieved content. Our work does not aim to enhance the reasoning capabilities of QA models themselves. Instead, our contributions are focused on improving document modeling during ingestion and leveraging that structure at inference to retrieve more relevant and complete evidence. The scope of this paper is therefore limited to retrieval improvements, not generative reasoning or answer synthesis.

Our framework relies on a powerful general-purpose LVLM to extract and structure document elements (e.g., text, tables, figures) during ingestion. Although manual inspection confirms that the extractive tasks, such as reading text from images or summarizing the content of tables and figures, were handled correctly in most cases, these models can still struggle with noisy inputs, complex layouts, or low-quality visuals. This reflects a broader trade-off between using a unified model that minimizes system complexity and integrating multiple specialized tools that may boost robustness but increase engineering overhead. Future work could explore modular alternatives tailored to specific document modalities when needed.

Acknowledgment

References

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, and 1 others. 2024. Phi-4 technical report. *arXiv preprint arXiv:2412.08905*.

Pravesh Agrawal, Szymon Antoniak, Emma Bou Hanna, Baptiste Bout, Devendra Chaplot, Jessica Chudnovsky, Diogo Costa, Baudouin De Monicault, Saurabh Garg, Theophile Gervet, and 1 others. 2024. Pixtral 12b. *arXiv preprint arXiv:2410.07073*.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, and 1 others. 2025. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*.

Aniket Bhattacharyya, Anurag Tripathi, Ujjal Das, Archan Karmakar, Amit Pathak, and Maneesh Gupta. 2025. Information extraction from visually rich documents using llm-based organization of documents into independent textual segments. *arXiv preprint arXiv:2505.13535*.

Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008.

Peter Baile Chen, Tomer Wolfson, Michael Cafarella, and Dan Roth. 2025. Enrichindex: Using llms to enrich retrieval indices offline. *arXiv preprint arXiv:2504.03598*.

Wenhu Chen, Hexiang Hu, Xi Chen, Pat Verga, and William W Cohen. 2022. Murag: Multimodal retrieval-augmented generator for open question answering over images and text. *arXiv preprint arXiv:2210.02928*.

Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, and 1 others. 2024. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24185–24198.

Jaemin Cho, Debanjan Mahata, Ozan Irsoy, Yujie He, and Mohit Bansal. 2024. M3docrag: Multimodal retrieval is what you need for multi-page multi-document understanding. *arXiv preprint arXiv:2411.04952*.

Chao Deng, Jiale Yuan, Pi Bu, Peijie Wang, Zhong-Zhi Li, Jian Xu, Xiao-Hui Li, Yuan Gao, Jun Song, Bo Zheng, and 1 others. 2024. Longdocurl: a comprehensive multimodal long document benchmark integrating understanding, reasoning, and locating. *arXiv preprint arXiv:2412.18424*.

André V Duarte, João Marques, Miguel Graça, Miguel Freire, Lei Li, and Arlindo L Oliveira. 2024. Lumberchunker: Long-form narrative document segmentation. *arXiv preprint arXiv:2406.17526*.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Manuel Faysse, Hugues Sibille, Tony Wu, Bilel Omrani, Gautier Viaud, Céline Hudelot, and Pierre Colombo. 2024. Colpali: Efficient document retrieval with vision language models. *arXiv preprint arXiv:2407.01449*.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2(1).

Ziyu Gong, Yihua Huang, and Chengcheng Mai. 2025. Mmrag-docqa: A multi-modal retrieval-augmented generation method for document question-answering with hierarchical index and multi-granularity retrieval. *arXiv preprint arXiv:2508.00579*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Tiezheng Guo, Qingwen Yang, Chen Wang, Yanyi Liu, Pan Li, Jiawei Tang, Dapeng Li, and Yingyou Wen. 2024. Knowledgenavigator: Leveraging large language models for enhanced reasoning over knowledge graph. *Complex & Intelligent Systems*, 10(5):7063–7076.

Haoyu Han, Yu Wang, Harry Shomer, Kai Guo, Jiayuan Ding, Yongjia Lei, Mahantesh Halappanavar, Ryan A Rossi, Subhabrata Mukherjee, Xianfeng Tang, and 1 others. 2024. Retrieval-augmented generation with graphs (graphrag). *arXiv preprint arXiv:2501.00309*.

Siwei Han, Peng Xia, Ruiyi Zhang, Tong Sun, Yun Li, Hongtu Zhu, and Huaxiu Yao. 2025. Mdocagent: A multi-modal multi-agent framework for document understanding. *arXiv preprint arXiv:2503.13964*.

Anwen Hu, Haiyang Xu, Jiabo Ye, Ming Yan, Liang Zhang, Bo Zhang, Chen Li, Ji Zhang, Qin Jin, Fei Huang, and 1 others. 2024. mplug-docowl 1.5: Unified structure learning for ocr-free document understanding. *arXiv preprint arXiv:2403.12895*.

De-An Huang, Subhashree Radhakrishnan, Zhiding Yu, and Jan Kautz. 2025. Frag: Frame selection augmented generation for long video and long document understanding. *arXiv preprint arXiv:2504.17447*.

Yulong Hui, Yao Lu, and Huanchen Zhang. 2024. Uda: A benchmark suite for retrieval augmented generation in real-world document analysis. *Advances in Neural Information Processing Systems*, 37:67200–67217.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Chelsi Jain, Yiran Wu, Yifan Zeng, Jiale Liu, Zhenwen Shao, Qingyun Wu, Huazheng Wang, and 1 others. 2025. Simpledoc: Multi-modal document understanding with dual-cue page retrieval and iterative refinement. *arXiv preprint arXiv:2506.14035*.

Boran Jiang, Yuqi Wang, Yi Luo, Dawei He, Peng Cheng, and Liangcai Gao. 2024. Reasoning on efficient knowledge paths: knowledge graph guides large language model for domain question answering. In *2024 IEEE International Conference on Knowledge Graph (ICKG)*, pages 142–149. IEEE.

Samira Ebrahimi Kahou, Vincent Michalski, Adam Atkinson, Ákos Kádár, Adam Trischler, and Yoshua Bengio. 2017. Figureqa: An annotated figure dataset for visual reasoning. *arXiv preprint arXiv:1710.07300*.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick SH Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pages 6769–6781.

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*, pages 39–48.

Yejin Kim, Eojin Kang, Juae Kim, and H Howie Huang. 2024. Causal reasoning in large language models: A knowledge graph approach. *arXiv preprint arXiv:2410.11588*.

Hugo Laurençon, Léo Tronchon, Matthieu Cord, and Victor Sanh. 2024. What matters when building vision-language models? *Advances in Neural Information Processing Systems*, 37:87874–87907.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916.

Wensheng Lu, Keyu Chen, Ruizhi Qiao, and Xing Sun. 2025. Hichunk: Evaluating and enhancing retrieval-augmented generation with hierarchical chunking. *arXiv preprint arXiv:2509.11552*.

Xing Han Lù. 2024. Bm25s: Orders of magnitude faster lexical search via eager sparse scoring. *arXiv preprint arXiv:2407.03618*.

Chuwei Luo, Yufan Shen, Zhaoqing Zhu, Qi Zheng, Zhi Yu, and Cong Yao. 2024. Layoutllm: Layout instruction tuning with large language models for document understanding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15630–15640.

Yubo Ma, Yuhang Zang, Liangyu Chen, Meiqi Chen, Yizhu Jiao, Xinze Li, Xinyuan Lu, Ziyu Liu, Yan Ma, Xiaoyi Dong, and 1 others. 2024. Mmlongbench-doc: Benchmarking long-context document understanding with visualizations. *Advances in Neural Information Processing Systems*, 37:95963–96010.

Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. 2022. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. *arXiv preprint arXiv:2203.10244*.

Minesh Mathew, Dimosthenis Karatzas, and CV Jawahar. 2021. Docvqa: A dataset for vqa on document images. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 2200–2209.

Hai-Toan Nguyen, Tien-Dat Nguyen, and Viet-Ha Nguyen. 2024. Enhancing retrieval augmented generation with hierarchical text segmentation chunking. In *International Symposium on Information and Communication Technology*, pages 209–220. Springer.

Mengjia Niu, Hao Li, Jie Shi, Hamed Haddadi, and Fan Mo. 2024. Mitigating hallucinations in large language models via self-refinement-enhanced knowledge retrieval. *arXiv preprint arXiv:2405.06545*.

Mateusz Płonka, Krzysztof Kocot, Kacper Hołda, Krzysztof Daniec, and Aleksander Nawrat. 2025. A comparative evaluation of the effectiveness of document splitters for large language models in legal contexts. *Expert Systems with Applications*, 272:126711.

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

Gaston Saux, Mary Anne Britt, Nicolas Vibert, and Jean-François Rouet. 2021. Building mental models from multiple texts: How readers construct coherence from inconsistent sources. *Language and Linguistics Compass*, 15(3):e12409.

Aditya Sharma, Michael Saxon, and William Yang Wang. 2024. Losing visual needles in image haystacks: Vision language models are easily distracted in short and long contexts. *arXiv preprint arXiv:2406.16851*.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2023. Replug: Retrieval-augmented black-box language models. *arXiv preprint arXiv:2301.12652*.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel M Ni, Heung-Yeung Shum, and Jian Guo. 2023. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. *arXiv preprint arXiv:2307.07697*.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, and 1 others. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Rubèn Tito, Dimosthenis Karatzas, and Ernest Valveny. 2023. Hierarchical multimodal transformers for multipage docvqa. *Pattern Recognition*, 144:109834.

Jordy Van Landeghem, Rubèn Tito, Łukasz Borchmann, Michał Pietruszka, Pawel Joziak, Rafal Powalski, Dawid Jurkiewicz, Mickaël Coustaty, Bertrand Anckaert, Ernest Valveny, and 1 others. 2023. Document understanding dataset and evaluation (dude). In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19528–19540.

Hengyi Wang, Haizhou Shi, Shiwei Tan, Weiyi Qin, Wenyuan Wang, Tunyu Zhang, Akshay Nambi, Tanuja Ganu, and Hao Wang. 2024a. Multimodal needle in a haystack: Benchmarking long-context capability of multimodal large language models. *arXiv preprint arXiv:2406.11230*.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

Yu Wang, Nedim Lipka, Ryan A Rossi, Alexa Siu, Ruiyi Zhang, and Tyler Derr. 2024b. Knowledge graph prompting for multi-document question answering. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 19206–19214.

Zhaowei Wang, Wenhao Yu, Xiyu Ren, Jipeng Zhang, Yu Zhao, Rohit Saxena, Liang Cheng, Ginny Wong, Simon See, Pasquale Minervini, and 1 others. 2025a. Mmlongbench: Benchmarking long-context vision-language models effectively and thoroughly. *arXiv preprint arXiv:2505.10610*.

Zhitong Wang, Cheng Gao, Chaojun Xiao, Yufei Huang, Shuzheng Si, Kangyang Luo, Yuzhuo Bai, Wenhao Li, Tangjian Duan, Chuancheng Lv, and 1 others. 2025b. Document segmentation matters for retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 8063–8075.

Xixi Wu, Yanchao Tan, Nan Hou, Ruiyang Zhang, and Hong Cheng. 2025. Molorag: Bootstrapping document understanding via multi-modal logic-aware retrieval. *arXiv preprint arXiv:2509.07666*.

Yifan Wu, Lutao Yan, Leixian Shen, Yunhai Wang, Nan Tang, and Yuyu Luo. 2024. Chartinsights: Evaluating multimodal large language models for low-level chart question answering. *arXiv preprint arXiv:2405.07001*.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2024. C-pack:

Packed resources for general chinese embeddings. In *Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval*, pages 641–649.

Yang Xu, Yiheng Xu, Tengchao Lv, Lei Cui, Furu Wei, Guoxin Wang, Yijuan Lu, Dinei Florencio, Cha Zhang, Wanxiang Che, and 1 others. 2020. Layoutlmv2: Multi-modal pre-training for visually-rich document understanding. *arXiv preprint arXiv:2012.14740*.

Zhengzhuo Xu, Sinan Du, Yiyan Qi, Chengjin Xu, Chun Yuan, and Jian Guo. 2023. Chartbench: A benchmark for complex visual reasoning in charts. *arXiv preprint arXiv:2312.15915*.

Jeff Yang, Duy-Khanh Vu, Minh-Tien Nguyen, Xuan-Quang Nguyen, Linh Nguyen, and Hung Le. 2025. Superrag: Beyond rag with layout-aware graph modeling. *arXiv preprint arXiv:2503.04790*.

Yuan Yao, Tianyu Yu, Ao Zhang, Chongyi Wang, Junbo Cui, Hongji Zhu, Tianchi Cai, Haoyu Li, Weilin Zhao, Zhihui He, and 1 others. 2024. Minicpm-v: A gpt-4v level mllm on your phone. *arXiv preprint arXiv:2408.01800*.

Shukang Yin, Chaoyou Fu, Sirui Zhao, Ke Li, Xing Sun, Tong Xu, and Enhong Chen. 2024. A survey on multimodal large language models. *National Science Review*, 11(12):nwae403.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, and 1 others. 2023. Siren's song in the ai ocean: a survey on hallucination in large language models. arxiv. *arXiv preprint arXiv:2309.01219*.

Yucheng Zhou, Zhi Rao, Jun Wan, and Jianbing Shen. 2024. Rethinking visual dependency in long-context reasoning for large vision-language models. *arXiv preprint arXiv:2410.19732*.

A Code Availability

The implementation of LAD-RAG is currently undergoing institutional review and legal clearance prior to public release. In the meantime, we provide full transparency to support reproducibility: the paper includes detailed descriptions of all components, comprehensive hyperparameter settings, and the full set of prompt templates used throughout the pipeline (see Appendix G). The codebase will be made publicly available upon approval, and the repository link will be added in the camera-ready version. For urgent reviewer needs, the code can be privately shared upon request.

B Dataset Details

MMLongBench-Doc. MMLongBench-Doc (Ma et al., 2024) targets long-context document understanding. It contains 1,082 expert-annotated questions over 135 lengthy PDF documents (average 47.5 pages, ~21k tokens per document). Evidence comes from multimodal sources, including text, images, charts, tables, and layout structure. Notably, 33% of questions require cross-page evidence. These properties make the benchmark especially challenging for retrieval completeness. The dataset is released under the Apache-2.0 license and is permitted for academic and research use.

LongDocURL. LongDocURL (Deng et al., 2024) integrates three task categories, understanding, reasoning, and locating, over 396 documents spanning ~33k pages. It provides 2,325 high-quality QA pairs, averaging 86 pages and ~43k tokens per document. Compared with earlier benchmarks (Ma et al., 2024; Van Landeghem et al., 2023; Tito et al., 2023), LongDocURL features the highest proportion of multi-page (52.9%) and cross-element (37.1%) questions, reflecting realistic challenges in handling heterogeneous evidence distributed across different pages and element types (e.g., paragraphs, tables, figures). The dataset is released under the Apache-2.0 license and is permitted for academic and research use.

DUDE. The Document Understanding Dataset and Evaluation (DUDE) (Van Landeghem et al., 2023) is a large-scale, multi-domain benchmark built from ~5k multi-page documents. It spans a wide spectrum of domains (medical, legal, technical, financial) and question types (extractive, abstractive, arithmetic, multi-hop, and non-answerable). On average, documents are 5.7 pages long with ~1,831 tokens. Although both extractive and abstractive questions may require evidence from multiple pages, only the extractive subset provides explicit annotations of the evidence pages. Within this subset, questions requiring multi-page evidence constitute roughly 1% of the total. The dataset is released under the CC BY 4.0 license and is permitted for academic and research use.

MP-DocVQA. MP-DocVQA (Tito et al., 2023) extends the original DocVQA dataset (Mathew et al., 2021) to multi-page documents. It comprises 46k questions over 5,928 documents, totaling 47,952 pages, approximately 8 pages per document. Although the documents span multiple

pages, each question is designed such that its supporting evidence is confined to a single page. As a result, the benchmark does not focus on multi-page evidence aggregation. Nevertheless, MP-DocVQA remains a valuable resource for evaluating retrieval performance in multi-page settings, particularly for assessing how well systems localize relevant information within longer documents. The dataset is released under the MIT license and is permitted for academic and research use.

C   Case Study

To illustrate the advantages of LAD-RAG's dynamic and layout-aware retrieval strategy, we present two case studies. These examples demonstrate how conventional RAG methods often fall short in document-centric tasks that require structural reasoning, and how LAD-RAG's combination of symbolic document graphs, semantic search, and dynamic control enables more complete and context-aware retrieval.

As shown in Figure 5, the task is to identify all charts in the report that compare the general public with Latinos/Hispanics. A conventional RAG pipeline performs semantic search using the full question string and retrieves a few charts that are semantically similar to the query. However, it misses many relevant charts, includes unrelated ones, and lacks a clear mechanism to determine how many results are needed to fully answer the question. This limitation stems from the fact that figure captions and labels are often brief, stylized, or ambiguous, making them difficult to match through semantic similarity alone.

LAD-RAG's dynamic retriever agent addresses these challenges by shifting from semantic to symbolic reasoning. It first filters the document graph to collect all nodes labeled as figures. It then contextualizes each node by examining surrounding elements such as section headers, nearby text, and related figure groups. This layout-aware process allows LAD-RAG to accurately determine whether a figure matches the query. As a result, it retrieves all relevant charts, including those distributed across pages or sharing titles. This example illustrates the advantages of dynamic control and symbolic reasoning in enabling more complete and precise retrieval for complex document-centric queries.

In Figure 6, we present a reference-based question that requires retrieving a multi-page appendix. A standard semantic retriever correctly identifies

the node titled "Appendix A: References" but fails to locate its continuation on the next page, which lacks semantic similarity to the query. As a result, critical evidence is missed.

LAD-RAG overcomes this by leveraging its symbolic document graph, which encodes layout continuity and section hierarchy. Its dynamic agent follows up on semantic search with graph-based contextualization when additional evidence may extend beyond the initially matched content. Because the document graph already groups nodes into coherent structures during ingestion (via community detection), the retriever can directly exploit these relationships at inference. This allows it to seamlessly retrieve all nodes belonging to the same section or evidence cluster, producing a more coherent and complete set of retrieved evidence.

Together, these two cases underscore the unique advantages of LAD-RAG. Rather than relying solely on embedding similarity, it dynamically decides when to use symbolic structure and how to combine it with semantic cues to retrieve comprehensive, contextually grounded evidence from visually complex documents.
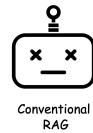
D   Additional Analysis on LAD-RAG's Retrieval Performance

Figure 7 provides a focused analysis of retrieval performance on multi-page evidence questions, which pose a greater challenge due to the need for cross-page contextual integration. We compare LAD-RAG against conventional top-k retrievers under varying retrieval depths. Our method consistently achieves higher recall with fewer irrelevant pages, highlighting its ability to prioritize semantically and structurally relevant content across document boundaries. Dotted horizontal lines show the minimum number of pages baseline methods must retrieve to match LAD-RAG's recall without requiring top-k tuning. These results underscore the importance of layout-aware, symbolically grounded retrieval strategies for handling complex, distributed information in long documents.

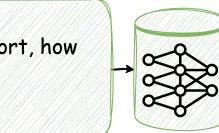## E   Additional Analysis on LAD-RAG's QA Performance

Table 3 provides a fine-grained breakdown of QA accuracy on MMLongBench-Doc, categorized by different evidence sources (e.g., layout, table, figure) and page types (e.g., single-page, multi-page). We observe consistent improvements across all types

Figure 5: Case study showing LAD-RAG retrieving all charts that compare the general public with Latinos/Hispanics. While a conventional semantic retriever fails to recall many relevant charts and includes irrelevant ones, LAD-RAG dynamically opts for the symbolic retrieval. It filters for all figure nodes and then contextualizes them using the surrounding layout to determine whether they match the query. This multi-step, graph-guided process enables accurate and exhaustive evidence collection.

when using LAD-RAG, with particularly large gains on layout-rich and chart-based questions, highlighting the model's ability to leverage positional and structural cues that are often missed by standard neural retrieval. Improvements are also notable in multi-page contexts, emphasizing the effectiveness of our symbolic document graph in modeling inter-page dependencies that neural retrievers typically fail to capture.

A similar pattern is observed in Table 4 for the LongDocURL benchmark. LAD-RAG consistently improves QA accuracy across models and across diverse question types, including understanding, reasoning, and location-based tasks. The gains span questions requiring different forms of document comprehension, from interpreting spatial layout to integrating multi-element evidence, highlighting the versatility of the retrieval strategy. These results suggest that LAD-RAG's benefits are not limited to specific evidence types or query styles, but generalize across a wide range of task demands in visually rich documents.

F   Latency Analysis

We also measure token generation across all calls. Figure 8 shows that more than 97% of the calls generate fewer than 100 tokens. This distribution highlights that the additional reasoning steps incurred by LAD-RAG are lightweight compared to the overall LLM inference cost. These results, alongside the main results presented in Section 4.4, demonstrate that our retrieval strategy introduces only minimal overhead relative to standard RAG pipelines, while delivering significantly higher retrieval completeness and QA accuracy.
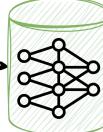
G   Implementation Details

We provide all relevant implementation details, including hardware setup, software environment, hyperparameter choices, and prompts used throughout the LAD-RAG framework. The symbolic document graph was implemented using the Python networkx library, which offers rich functionality for representing, analyzing, and traversing graph

Q: *Among all 12 references in this report, how many are from its own research center?*

LAD-RAG

Use semantic search to find nodes related to "references" or "citations" in the document `<code> do_semantic_search ("references citations", doc) </code>`

Appendix A: References
Page 22

Contextualize the "Appendix A: References" section header node to find related content that might list the references. `<code> get_community_for_node("page_22-obj_002", doc_graph) </code>`

Appendix A: References
Page 22

Appendix cont.
Page 23

Figure 6: Case study showing LAD-RAG retrieving a multi-page reference section. While semantic search finds only the first page of references, it misses the continuation due to weak semantic overlap. LAD-RAG dynamically switches to graph-based contextualization to recover all structurally related nodes, enabling full evidence coverage.
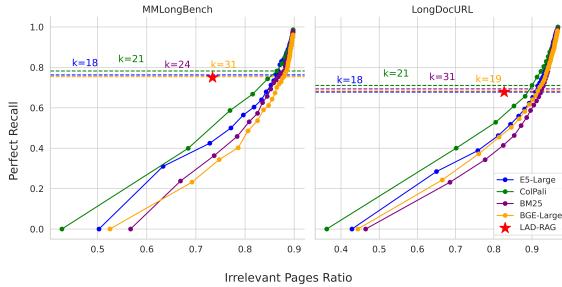

Figure 7: Retrieval performance on multi-page evidence questions. Our framework is compared to baseline retrievers across varying top-k values. Dotted horizontal lines indicate how many pages each baseline must retrieve to match the recall of LAD-RAG without top-k tuning.
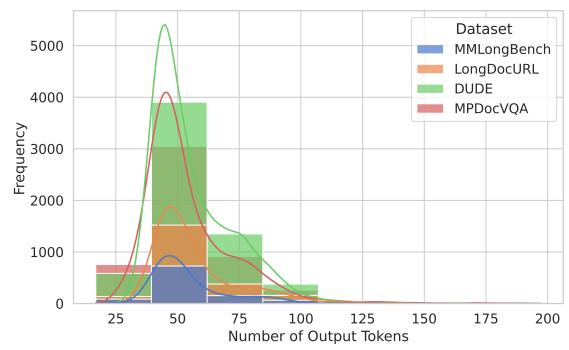

Figure 8: Distribution of tokens generated per query across all LLM calls. The vast majority of queries use fewer than 100 tokens, indicating that the additional reasoning overhead introduced by our dynamic retrieval is lightweight in practice.

structures. For model inference, we used the vLLM library (version 0.9.2), which enables fast and efficient LLM serving with minimal overhead. LAD-RAG experiments were conducted on 4 NVIDIA A100 GPUs. The software environment included Python 3.10.12 and PyTorch version `2.7.0+cu126`.

G.1 Hyperparameters

We used temperature $= 0$ for all prompting steps involving the LVLM, including (i) document graph construction (both node and edge generation), (ii) inference-time retrieval via the agent, and (iii) the QA stage. The maximum token limit was set to 8192 for all document graph construction steps, such as page-level element extraction, memory updates, and relation extraction, and 2048 for all QA-related evaluations. For the retriever agent, we set the maximum number of interaction rounds (conversations) to 20. Furthermore, for the baseline

retrievers, we evaluate performance across k values up to the point of perfect recall. On average, this occurs at $k=94$ for MMLongBench, $k=65$ for LongDocURL, $k=17$ for DUDE, and $k=10$ for MP-DocVQA.

For image rendering, we used the `PyMuPDF` library to convert each PDF page into an image with a resolution of 300 DPI. During inference, we preserved the original image resolution when GPU memory allowed; in constrained scenarios, we scaled the resolution down to 50% or, in rare cases, to 20% of the original size. We manually verified that models could still correctly extract textual and visual information from these downsampled images, ensuring that no critical semantic or layout content was lost.

Model	Retrieval	Evidence Source					Evidence Page		
		Layout	Text	Table	Figure	Chart	Single	Multi	UNA
InternVL2-8B	Ground-Truth	0.385	0.427	0.315	0.380	0.339	0.506	0.250	0.052
	retrieving@5	0.256	0.298	0.266	0.199	0.295	0.372	0.164	**0.054**
	retrieving@10	0.274	0.348	0.280	0.236	0.307	0.395	0.208	0.051
	topk-adjusted	0.291	0.309	0.228	0.214	0.308	0.365	0.212	0.051
	LAD-RAG	**0.361**	**0.408**	**0.340**	**0.321**	**0.363**	**0.495**	**0.242**	0.053
Pixtral-12B-2409	Ground-Truth	0.513	0.461	0.431	0.398	0.412	0.537	0.343	0.065
	retrieving@5	0.248	0.343	0.321	0.250	0.288	0.395	0.200	0.065
	retrieving@10	0.294	0.239	0.280	0.240	0.226	0.327	0.178	0.063
	topk-adjusted	0.308	0.326	0.330	0.269	0.282	0.394	0.213	0.066
	LAD-RAG	**0.336**	**0.398**	**0.387**	**0.316**	**0.330**	**0.462**	**0.261**	**0.070**
Phi-3.5-Vision	Ground-Truth	0.376	0.416	0.300	0.366	0.301	0.492	0.227	0.634
	retrieving@5	0.248	0.324	0.259	0.243	0.254	0.399	0.192	0.616
	retrieving@10	0.256	0.338	0.280	0.209	0.273	0.381	0.189	0.612
	topk-adjusted	0.267	0.351	0.300	0.244	0.290	0.400	0.199	0.611
	LAD-RAG	**0.299**	**0.364**	**0.304**	**0.279**	**0.324**	**0.414**	**0.202**	**0.632**
GPT-4o-200b-128	Ground-Truth	0.597	0.628	0.638	0.536	0.616	0.693	0.565	**0.206**
	retrieving@5	0.378	0.469	0.500	0.395	0.435	0.607	0.303	0.205
	retrieving@10	0.424	0.495	0.560	0.426	0.486	0.637	0.372	0.190
	topk-adjusted	0.480	0.510	0.568	0.394	0.528	0.629	0.409	0.202
	LAD-RAG	**0.500**	**0.576**	**0.607**	**0.478**	**0.534**	**0.676**	**0.450**	0.205

Table 3: Breakdown of QA accuracy on MMLongBench-Doc. Best values per model group (excluding GT) are **boldfaced**.

G.2 Prompts

This subsection covers all prompts used with the LVLM across different components of the LAD-RAG framework. This includes prompts for generating document graph nodes and edges during ingestion, as well as the prompts used by the inference-time LLM agent to query the neuro-symbolic index and retrieve contextually relevant evidence to answer questions.

Document graph nodes extraction. We extract layout-aware objects from each document page using a specialized prompt designed to capture rich structural and semantic attributes. The full prompt can be found in Figure 9.

Running memory construction To build a persistent and structured working memory across multi-page documents, we use a specialized prompt that summarizes, links, and updates previously extracted elements while maintaining coherence and document flow. This prompt helps the system accumulate context and track evolving sections or entities across pages. The complete prompt is shown in Figure 10.

Document graph generation. We provide the full prompt used to construct the document graph during ingestion, with a focus on modeling cross-page structure and dependencies. This prompt guides the model to maintain a structured working memory across pages, update hierarchical and semantic state, and extract relationships that connect document elements (e.g., paragraphs, tables, figures) across page boundaries. It instructs how to represent elements as graph nodes and link them using edges encoding layout-aware, semantic, and interpretive relationships spanning multiple pages. The full text of the prompt is shown in Figure 11.

Retriever agent. We show the full prompt used to guide the retriever agent during inference. This agent is responsible for orchestrating multiple retrieval rounds based on partial observations and evolving context. The prompt describes how the agent should form hypotheses, refine queries, decide when to stop, and integrate symbolic and semantic cues to gather the most relevant evidence. See Figure 12 for the complete prompt.

Model	Retrieval	Evidence Source				Task Type		
		Layout	Text	Table	Figure	Und	Rea	Loc
InternVL2-8B	Ground-Truth	0.507	0.719	0.608	0.670	0.724	0.572	0.486
	retrieving@5	0.296	0.526	0.420	0.479	0.583	0.435	0.184
	retrieving@10	0.311	0.558	0.442	0.483	0.597	0.425	0.215
	topk-adjusted	0.288	0.557	**0.486**	0.536	0.547	0.443	0.295
	LAD-RAG	**0.319**	**0.571**	**0.486**	**0.556**	**0.618**	**0.625**	**0.297**
Pixtral-12B-2409	Ground-Truth	0.613	0.688	0.589	0.614	0.660	0.606	0.601
	retrieving@5	0.405	0.482	0.392	0.412	0.478	0.389	0.366
	retrieving@10	0.417	0.478	0.369	0.396	0.471	0.399	0.362
	topk-adjusted	0.433	0.504	0.418	0.427	0.502	0.418	0.386
	LAD-RAG	**0.462**	**0.553**	**0.490**	**0.475**	**0.559**	**0.469**	**0.432**
Phi-3.5-Vision	Ground-Truth	0.608	0.689	0.596	0.581	0.711	0.435	0.596
	retrieving@5	0.416	0.561	0.408	0.419	0.599	0.415	0.283
	retrieving@10	0.412	0.568	0.411	0.441	0.607	0.415	0.279
	topk-adjusted	0.415	0.549	0.404	0.432	0.606	0.318	**0.298**
	LAD-RAG	**0.425**	**0.577**	**0.428**	**0.459**	**0.619**	**0.425**	0.291
GPT-4o-200b-128	Ground-Truth	0.586	0.777	0.712	0.742	0.797	0.728	0.551
	retrieving@5	0.444	0.670	0.613	0.621	0.680	0.599	0.419
	retrieving@10	0.474	0.708	0.635	0.631	0.714	0.620	0.452
	topk-adjusted	0.532	**0.720**	0.655	0.680	0.732	0.619	0.511
	LAD-RAG	**0.536**	0.718	**0.671**	**0.709**	**0.735**	0.622	**0.528**

Table 4: Breakdown of QA accuracy on `LongDocURL`. Best values per model group (excluding `GT`) are **boldfaced**.

Prompt used for document graph node extraction

```
You are a document/image analysis assistant. The user is uploading a scanned or rendered page from a PDF file or it can even
↪  be an image. Your task is to extract a structured JSON representation of the page/image, breaking it down into individual,
↪  self-contained objects such as paragraphs, titles, section headers, tables, and figures.
For each detected object, extract the following fields:
- `type`: One of [title, section_header, paragraph, figure, table, footnote, metadata, etc.]
- `content`: The full raw content of the object. This must include **all information visible** in the element,
  whether it's a paragraph, table, or chart. For tables or figures, include captions, all data values, axis labels,
  legend descriptions, etc.
- `title_or_heading`: A short heading that describes what this object is about.
- `position_on_page`: Where this object appears (e.g., 'top-left', 'center-bottom', 'right margin').
- `layout_relation`: How this object is situated relative to other elements (e.g., 'below the main title',
  'next to bar chart', 'in footer section').
- `summary`: A **complete and self-contained** summary of the object. Summarize all meaningful information in the object.
  For example, for tables or charts, describe the key patterns, values, and implications clearly. The summary should be
  useful even when this object is retrieved alone in a retrieval-augmented generation (RAG) system.
- `visual_attributes`: Font size, emphasis (bold/italic), color, spacing, etc., if observable.
- `page_metadata`: Metadata such as `document_title`, `page_number`, `watermark`, `source_url`, if available.
- `object_id`: A unique ID (e.g., '{}-obj_003') that can be used to reference this object in a downstream system.
  All the IDs should start with the prefix '{}'. DO NOT ADJUST THIS PREFIX WITH THE PAGE NUMBER ON THE PAGE AND JUST USE THE
↪  PREFIX GIVEN TO YOU.
- (optional) `content_type`: A semantic tag like 'contact_info', 'citation', 'header', 'statistical_summary', etc., if
↪  applicable.
- (optional) `document_context`: A brief reference to the broader document or section context (e.g.,
  'From Pew Research Center report on U.S. immigration opinion, May 2018').
Important:
- Each object must be maximally self-contained and descriptive.
- The `content` field should include all information that appears in the visual object.
- The `summary` should explain the meaning, relevance, and insight of the object, not just repeat its text.
- Your output must be a **JSON list**, where each item is an object containing the fields listed above.
- If any field is unknown or unavailable, return null.
- The result must be optimized for downstream use in a RAG system.
```

Figure 9: Prompt used for document graph node extraction.

Prompt used for running memory construction

```
You are analyzing the structure of a multi-page document. Your task is to decide how to update the document's `section_queue`
↪  based on a list of new section-like objects detected on the current page.

The `section_queue` represents the current hierarchical structure of the document. It is a list that moves from the top-level
↪  section to deeper subsections. Each element is either:
- a single dictionary with `text` and `object_id` representing a section/subsection.

Each dictionary has the form:
- `"text"`: the section or subsection title
- `"object_id"`: the ID of the corresponding object

Now, for the current page, you are given `candidate_sections`, a list of newly extracted section-like objects. Your job is to
↪  determine:
- If these candidates are **new subsections** under the current section (append them).
- If the new candidates are semantically on the same levels as the leaf already in the current hierarchy, truncate the tree
↪  until you get to a broader topic and then insert them. Sometimes you might need to remove all the topic and start from
↪  scratch, which is OK.
- Or if they are just **continuing** the existing structure (no change).

You should use **semantic understanding** of the section titles to determine structural relationships — for example, whether
↪  "Threats to Democracy" fits under "Challenges in Governance" or signals a new section.

---

**CURRENT SECTION_QUEUE:**
{json.dumps(working_memory.get('section_queue', working_memory), indent=2)}

**CANDIDATE SECTIONS FROM PAGE:**
{json.dumps(candidate_section_objects, indent=2)}

Return only a JSON object with the updated section queue:
```json
{{
  "section_queue": [
    {{ "text": ..., "object_id": ... }},


    ...
    [{{ "text": ..., "object_id": ... }}, {{ "text": ..., "object_id": ... }}],
  ]
}}
Important rules:
- Only return the final updated section_queue JSON — no explanations or extra texts.
- It SHOULD NEVER happen that objects lower in the hierarchy and closer to the leaf would come from pages prior to the parent
↪  objects (this can be inferred from the object ids: ...page_{{ page number}}...). If you're updating the section_queue like
↪  that, something is wrong; Do it again. objects closer to the leafs are both structurally (page-wise) and also semantically
↪  following parent objects and NOT the other way around.
```

Figure 10: Prompt used for constructing and updating the running memory across document pages.

Prompt used for document graph construction

```
You are continuing the analysis of a document with multiple pages. Below is the content extracted from the current page,
↪   including structured objects and intra-page relationships, as well as the actual document page which is given to you.

Use this information along with your memory of prior pages and the structural information obtained so far to:

- **Update the working memory**. The working memory should maintain both semantic and structural state across pages, including:
  · **section_queue**: hierarchical list of sections and subsections, possibly including lists of sibling sections at the
  ↪   same level. Do **not** change this unless explicitly asked; it has already been updated.
  · **active_entities**: a list of important entities that persist across pages, each with `text` and associated `object_id(s)`
  · **semantic_topics**: any core topics being discussed that continue across pages
  · **unresolved_objects**: any objects (e.g., tables, figures) that are referred to but not fully explained yet (with
  ↪   `object_id`)

- **Identify cross-page relationships** between this page and earlier pages. This includes:

  1. **Hierarchical references**:
     - All extracted objects from the current page (e.g., paragraphs, figures, tables) should be **explicitly connected** to
     ↪   the most relevant section or subsection from the current `section_queue`, prioritizing the lower leaf levels.
     - If there are multiple valid target sections or subsections (e.g., leaf or its parent in the hierarchy), use both.
     - Represent these using:
       {{
         "from_object": "object_id_of_child",
         "to_object": "object_id_of_section",
         "type": "is_part_of_section"
       }}

  2. **Interpretive links**:
     - For example, if a paragraph on this page explains a table from a previous page, return:
       {{
         "from_object": "paragraph_obj_id",
         "to_object": "table_obj_id",
         "type": "explains"
       }}

  3. **Object-level semantic links**:


     - Link the current page's objects to earlier objects referenced in working memory using semantic types such as:
       `"continues"`, `"references"`, `"summarizes"`, `"updates"`, ...
     - These links should be grounded in the objects tracked in the working memory (e.g., those listed in `active_entities`,
     ↪   `semantic_topics`, and `unresolved_objects`).

CURRENT PAGE OBJECTS:
{extracted_objects_text}

CURRENT PAGE RELATIONSHIPS:
{extracted_relations_text}

CURRENT WORKING MEMORY:
{json.dumps(working_memory, indent=2)}


Return a JSON object with the following fields:
- "updated_memory": updated version of the working memory
- "cross_page_relationships": a list of new relationships across pages in the format:
  {{
    "from_object": "object_id_of_child_or_reference",
    "to_object": "object_id_of_parent_or_target",
    "type": "relationship_type"
  }}
```

Figure 11: Prompt used for document graph construction during ingestion.

```
<task_description>
You are an agent designed to retrieve evidence from a visually-rich document using a structured document graph.
Your goal is to dynamically determine and retrieve the best set of evidence nodes based on the question.
Do NOT try to filter too much based on small details and as much as possible try to prioritize recall over precision. for
↪  instance, if a question is asking about all the charts or figures with certain characteristic, get all the figures and
↪  return them as when you're doing extra filtering you might lose important evidences.
You can request code to operate over either:
    - A semantic similarity index (for conceptually close matches),
    - A graph-based document structure (for layout-aware or structural reasoning),
    - Or a combination of both.
The retrieval process is iterative. First, create a high-level plan of what operations are required (numbered), and then for
↪  each step, generate a Python code snippet. I will execute each code snippet and return the results, until you determine
↪  that sufficient evidence has been retrieved.
</task_description>
<data_description>
Each evidence comes from a node in an undirected NetworkX document graph with the following structure; Not all keys are
↪  guaranteed to exist—code should handle missing fields gracefully.
<node>
    <type>One of: aggregated_section, paragraph, section_header, title, figure, metadata, table, footnote, footer, list, note,
    ↪  quote, citation, link, contact_info, code, warning, map, page_number, code_block</type>
    <content>Full text of the node</content>
    <summary>Natural language summary of content</summary>
    <object_id>Unique node identifier which can have the format [main folder]/[document]/page_[page number]-obj_[object
    ↪  number];
    </object_id>
</node>
</data_description>
<available_operations>
You can use the following operations in your plan:
1. Provide one string query to retrieve top-k semantically similar nodes.
2. Use structural filters to directly select nodes based on their attributes.
3. Given a node's object_id, return related content based on graph structure. This operation provides contextualization for
↪  nodes, so try to use it as much possible.
</available_operations>
<task_flow>


1. Read the <question> and understand what kind of evidence it requires (e.g., something more semantic or layout-related,
↪  e.g., figure? table? all sections on a page?).
2. Propose a <plan> with numbered steps and a short description of each.
3. For each step in the plan, generate valid Python code.
4. Wait for me to execute and return the result.
5. Iterate as needed until all relevant evidence is retrieved and do not filter too much; again recall is more important than
↪  precision.
7. Finish with a list of selected node IDs in <code></code> and <step_keyword>DONE</step_keyword>.
</task_flow>
<output_format>
Always start with:
    <plan>...</plan>
Then, for each step, which is also a turn in our conversation:
    <step>
        <description>Short goal of this step</description>
        <step_keyword>from the set {{semantic_search,graph_filter,graph_contextualize,DONE}}</step_keyword>
        <code>
            # Python code here
        </code>
    </step>
**for semantic_search, you have access to the function do_semantic_search(query, {}) that gets a query in string and the
↪  pdf_name that you do NOT need to change, and retrieves the most relevant nodes matching wrt to their summary and content
↪  for that string. you can do this multiple times if you think you need to retrieve the semantically similar items for
↪  multiple queries that are part of the main query. Be sure to only give me the function call; I will put the result of the
↪  function call in a variable and give it back to you myself.
**for graph_filter, you have access to a doc_graph that is basically the networkX object we talked about. You should give me
↪  the script that works with this doc_graph and its content is a list of nodes after filtering; Be sure to give me a
↪  expression not a statement; for instance [(node_id, node) for node_id, node in doc_graph.nodes(data=True) if ...].
**for graph_contextualize, you have access to the function get_community_for_node(node_id, doc_graph) that basically gets all
↪  the nodes that are in the community of that specific node. Be sure to only give me the function call; I will put the
↪  result of the function call in a variable and give it back to you myself.
</output_format>
<instructions>
You are now ready to begin. Think step by step and act like a smart retrieval agent.
Respond by first outputting a <plan> based on the following question: "{}"
</instructions>
```

Figure 12: Prompt used to guide the retriever agent during dynamic iterative evidence retrieval.