LILaC: Late Interacting in Layered Component Graph for Open-domain Multimodal Multihop Retrieval

Anonymous ACL submission

Abstract

Multimodal document retrieval aims to retrieve query-relevant components from documents composed of textual, tabular, and visual elements. An effective multimodal retriever needs to handle two main challenges: (1) mitigate the effect of irrelevant contents caused by fixed, single-granular retrieval units, and (2) support multihop reasoning by effectively capturing semantic relationships among components within and across documents. To address these challenges, we propose LILaC, a multimodal retrieval framework featuring two core innovations. First, we introduce a *layered component graph*, explicitly representing multimodal information at two layers—each representing coarse and fine granularity—facilitating efficient yet precise reasoning. Second, we develop a *late-interaction-based subgraph retrieval* method, an edge-based approach that initially identifies coarse-grained nodes for efficient candidate generation, then performs fine-grained reasoning via late interaction. Extensive experiments demonstrate that LILaC achieves state-of-the-art retrieval performance on four out of five benchmarks, notably without additional fine-tuning.

1 Introduction

Multimodal retrieval is a rapidly advancing research area, crucial for enhancing modern information retrieval systems (Li et al., 2022a, 2023; Radford et al., 2021). Early studies primarily focused on multimodal component retrieval, where components such as text, tables, and images had limited or no explicit relationships (Talmor et al., 2021; Chang et al., 2022; Li et al., 2022b). Recently, however, there has been an emerging shift toward open-domain multimodal document retrieval, where closely related components of various modalities are grouped together as a unified document, such as webpages or PDFs (Yu et al., 2024; Cho et al., 2024). Such multimodal documents can be viewed as collections of potentially interconnected components (e.g., via hyperlinks as shown with Taj Mahal in Figure 1), each belonging to one of multiple modalities, including text, tables, or images.

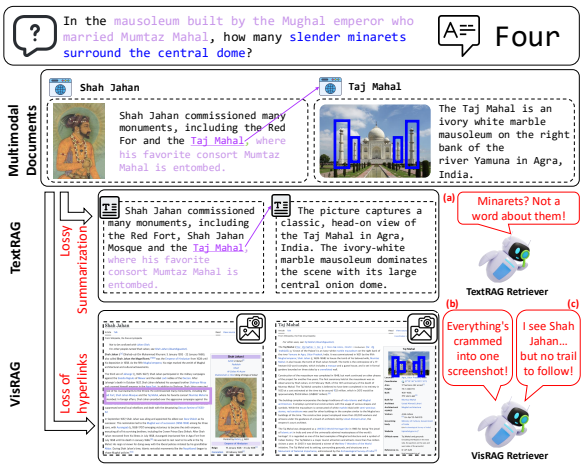


Figure 1: Challenges of TextRAG approaches and VisRAG approaches. (a) Incorrect summarization may result in possible information loss in TextRAG. (b) Insufficient retrieval granularity in VisRAG. (c) Limited multihop reasoning due to loss of links in VisRAG.

Recent approaches in multimodal document retrieval have increasingly adopted *VisRAG*-based methodologies, which unify diverse modalities by treating them primarily as visual content, typically represented through screenshots such as a page of a PDF file (Yu et al., 2024; Faysse et al., 2024; Cho et al., 2024). By casting multimodal retrieval as essentially an image retrieval problem, these methods leverage advanced vision-based embedding models to preserve multimodal information.

This paradigm emerged largely as a response to the limitations of earlier *TextRAG*-based approaches, which predominantly relied on textual retrieval by converting visual data into textual summaries (Yu et al., 2023b; Asai et al., 2023; Yan et al., 2024; Yang et al., 2023; Yu et al., 2023a; Luo et al., 2023). Although effective in leverag-

ing mature text retrieval systems, these methods inherently struggled to represent visual content adequately, resulting in potential information loss and reduction in retrieval effectiveness. For example, in Figure 1, the textual summary of the `Taj Mahal`'s image omits the word `minarets`, which was crucial for answering the query in this context.

Despite their conceptual advances, current multimodal retrieval approaches, including VisRAG, still face two crucial limitations:

(1) Insufficient consideration of retrieval granularity. Effective retrieval demands explicitly setting an optimal granularity of information representation (Chen et al., 2024). Existing VisRAG methods, however, typically adopt a fixed, single-granular approach—generally at the full-page screenshot level—which may include multiple components irrelevant to the query. Empirically, we observed that a single screenshot typically comprises an average of three distinct components. Consequently, the portion of query-relevant information within each screenshot is relatively small, inevitably leading to diminished embedding quality and retrieval effectiveness. Thus, granularity-aware retrieval remains largely unaddressed within multimodal document retrieval settings. For example, in Figure 1(b), VisRAG struggles because the query-relevant information constitutes only a small portion of the screenshot's content.

(2) Limited capability for multihop reasoning. Multimodal document retrieval inherently requires reasoning about complex intra- and inter-document relationships among components. Effective multihop reasoning critically depends on capturing these relationships, as within-document retrieval often necessitates integrating complementary information distributed across multiple modalities to fully represent an entity. Likewise, inter-document retrieval typically demands traversing semantic connections between related documents. Existing VisRAG-based approaches, however, independently embed and retrieve individual screenshots via nearest-neighbor search, thereby overlooking essential interdependencies among components. Moreover, these methods disregard inherent structural connections within the same document, such as associations among screenshots originating from the same page or hyperlinks explicitly linking different components. Although some multimodal component retrieval methods have introduced multihop reasoning capabilities (Yang et al., 2023), they largely focus on distractor-based closed-domain settings and rely heavily on online reasoning with Large Language Models, significantly limiting their generalization to open-domain multimodal document retrieval scenarios. For instance, in Figure 1(c), VisRAG struggles with multihop reasoning because it does not utilize the structural link from `Shah Jahan` to `Taj Mahal`.

To address the challenges, we propose `LILaC`, an effective multimodal retrieval approach with two novel ideas:

(1) Layered component graph construction. We first represent the multimodal document corpus as a layered component graph, explicitly designed to capture multimodal information at two distinct granularities. This layered graph structure leverages edges to explicitly encode relationships among components within and across documents, thus inherently facilitating effective multihop reasoning. Additionally, we utilize a layered representation, enhancing retrieval efficiency and effectiveness. The coarse-grained layer—where textual content is represented as paragraphs, tables as whole entities, and images in their entirety—provides contextual understanding suitable for broad candidate generation. While in the fine-grained layer—where paragraphs are extracted into sentences, tables into discrete rows, and images into detected visual objects—enables precise reasoning by decomposing content into finer units. Edges in the coarse-grained layer capture semantic associations among components, while edges connecting coarse-grained nodes to their fine-grained subcomponents represent hierarchical containment relationships.

(2) Late-interaction-based subgraph retrieval in layered graph. At online time, `LILaC` retrieves a query-relevant subgraph from the layered component graph. A key challenge in this step is the combinatorial explosion of candidate subgraphs, resulting from the extensive number of nodes and edges distributed across both granularity layers (Hu et al., 2024). To efficiently manage this complexity, we propose a traversal-based subgraph retrieval method on the layered component graph. Specifically, we first decompose the original query to identify an initial candidate node set at the coarse-grained layer. We then iteratively perform beam search by traversing connected edges from these initial candidates, dynamically computing relevance scores at each step. Crucially, since explicitly computing scores for all potential edges would be computationally prohibitive, we leverage the layered structure of both the graph and query de-

composition. In particular, edge scores are computed dynamically via late interaction between the fine-grained subqueries and the fine-grained nodes associated with each candidate edge, effectively utilizing node-level embeddings.

In summary, we make three key contributions: (1) We introduce a layered graph structure capturing multimodal documents at dual granularities, effectively supporting multihop reasoning. (2) We propose an efficient yet effective subgraph retrieval method leveraging late interaction between decomposed queries and fine-grained components. (3) Extensive experiments demonstrate that our approach achieves state-of-the-art retrieval accuracy on four out of five benchmarks, notably using only pretrained models without additional fine-tuning.

2 Preliminary

In this paper, we address multimodal document retrieval, defined as the task of retrieving a ranked list of multimodal components relevant to a given natural language query. Formally, a retrieval corpus $\mathcal{D}$ comprises a collection of multimodal documents $\{D_1, D_2, \ldots, D_{k_{doc}}\}$. Each multimodal document $D = [C_1, \ldots, C_{k_{comp}}]$ is a sequence of multimodal components. A multimodal component C may belong to one of three distinct modalities

- *Paragraph* P: a sequence of tokens, forming an unstructured text segment.
- *Table* T: a structured matrix with rows T_i indexed by row number i.
- *Image* I: a tensor $I \in \mathbb{R}^{w \times h \times a}$, with w, h, and a denote the width, height and the number of channels, respectively.

Given a natural language query Q, a retrieval corpus $\mathcal{D}$ and a link mapping $\mathcal{L}$, the retrieval task aims to produce a ranked list of components $\mathcal{R} = [C_1, \ldots, C_{n_{ret}}]$. The goal is for the ranked list $\mathcal{R}$ to contain the ground truth set of relevant components $C_{gt_1}, \ldots, C_{gt_r}$.

The link mapping $\mathcal{L} = \mathcal{C} \rightarrow \mathcal{D}$ represents the association or hyperlink relationships between individual components C and their respective multimodal documents D, similar to hyperlinks commonly used in webpages and PDF files.

3 Related Work

3.1 Multimodal Document Retrieval

Early multimodal retrieval methods primarily used a *text-centric* strategy, converting all components—paragraphs, tables, and figures—into plain text, thus losing essential visual cues (Yang et al., 2023; Yu et al., 2023a; Luo et al., 2023). Later approaches maintained separate embedding spaces for text and images, encoding each modality independently and merging their scores heuristically (Mei et al., 2025; Riedler and Langer, 2024). However, these methods struggle with reasoning across modalities due to disjoint embeddings.

Recent work pushes modality unification a step further through **VisRAG** pipelines: documents are rasterized into page- or region-level screenshots, so that paragraphs, tables, and images alike are embedded in a single *visual* space. `VisRAG` demonstrates end-to-end vision-based retrieval–augmented generation, while `ColPali` introduces a late-interaction vision–language model that produces multi-vector page embeddings. Despite their strengths, VisRAG approaches inherit some limitations. (i) **Fixed granularity:** retrieval granularity is fixed as full-page screenshots, which may contain query-irrelevant context. (ii) **Limited multihop reasoning:** current pipelines treat each screenshot independently, ignoring the dependencies between components.

3.2 Granularity of Retrieval

Previous studies have explored retrieval granularity across various modalities. In text retrieval, `DenseXRetrieval` demonstrates improved retrieval accuracy using finer sentence- and proposition-level units (Chen et al., 2024). `Mix-of-granularity` dynamically selects the optimal granularity tailored to each query (Zhong et al., 2024), while `RAPTOR` starts from sentences and recursively clusters and summarizes them into coarser units (Sarthi et al., 2024). For table modality, OTT-QA segments tables into header-plus-row units for targeted row-level retrieval (Herzig et al., 2021). However, granularity in multimodal document retrieval remains largely unexplored.

3.3 Multimodal Embedder Models

Recently, multimodal embedders and their corresponding benchmarks (Jiang et al., 2024; Wei et al., 2024) have emerged as active research areas due to the limitations of traditional uni- or cross-modal embedders in dynamic retrieval scenarios. Unlike conventional unimodal embedders, multimodal approaches specifically address dynamic settings characterized by retrieval tasks guided by explicit *modality instructions*. Advanced models such as `MMEmbed`, `UniME`, and `mmE5` leverage sophis-

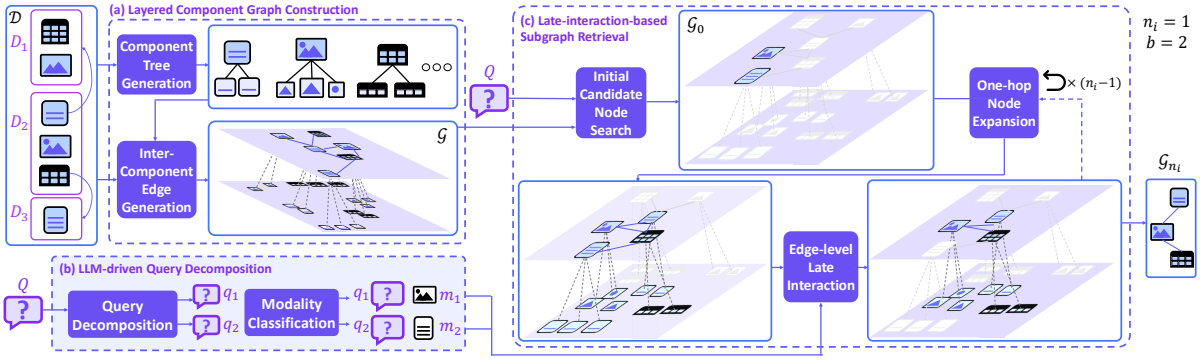


Figure 2: Overview of LILaC. (a) A layered component graph is constructed by organizing multimodal documents into coarse- and fine-grained layers. (b) The query is decomposed, followed by modality classification for each subquery. (c) LILaC dynamically retrieves a query-relevant subgraph through iterative beam-search traversal.

ticated multimodal language models along with modality-specific fine-tuning, significantly improving retrieval performance under clear modality instructions (Lin et al., 2024; Gu et al., 2025; Chen et al., 2025). However, existing multimodal embedders predominantly focus on training at the component level, leaving the effective use of these models for multimodal document retrieval largely unexplored. Furthermore, scenarios involving retrieval tasks without explicit instructions or with ambiguous contexts have yet to be thoroughly investigated.

4 Proposed Method

We propose LILaC, a novel retrieval algorithm utilizing a layered component graph and traversal method to retrieve a query-relevant subgraph. As shown in Figure 2, it consists of two stages: (i) **Layered Graph Construction** organizes multimodal documents into a layered component graph with explicit intra- and inter-document edges. (ii) **Late-interaction-based Subgraph Retrieval** iteratively traverses the layered graph in an edge-wise manner. To score an edge using node-level embeddings, it uses late interaction between the decomposed subqueries and low-layer subcomponents of an edge.

4.1 Layered Component Graph Construction

In the offline phase, LILaC constructs a layered graph structure $\mathcal{G}$, called the *layered component graph*, from the multimodal document set $\mathcal{D}$ and the associated link mapping $\mathcal{L}$. This graph comprises two distinct layers explicitly designed to represent semantic relationships among multimodal components, offering two primary advantages. First, the top layer supports multihop retrieval by explicitly modeling relationships between components and documents, enabling identification of relevant contexts. Second, the lower layer facilitates precise, fine-grained reasoning by further decomposing components into finer *subcomponents*, thus providing detailed context for accurate retrieval.

Definition 1 (Subcomponent)**.** *Let C be a multimodal component. A* subcomponent $c \in \mathcal{S}(C)$ *is defined in a modality-specific manner:*

- ***Paragraph.*** *For a paragraph $P = [p_1, \ldots, p_{k_{sent}}]$ consisting of sentences, each sentence p_j is a subcomponent.*
- ***Table.*** *Let $T = [T_0; T_1; \ldots; T_{k_{row}}]$ where T_0 is the header row. For every data row T_i $(1 \le i \le k_{row})$, the two-row segment $t_i = [T_0; T_i]$ is a subcomponent.*
- ***Image.*** *Given an image tensor $I \in \mathbb{R}^{w \times h \times a}$ and an object detector that returns a bounding box (x_1, y_1, x_2, y_2), the corresponding patch*
$$i = I[x_1 : x_2,\ y_1 : y_2,\ :]$$
is a subcomponent.

Definition 2 (Layered Component Graph)**.** *We define a layered component graph as $\mathcal{G} = (V, E, \lambda, \tau)$, where V is a set of vertices. A vertex v belongs to one of the two layers, determined by the layer map $\lambda : V \to \{0, 1\}$, where 0 and 1 corresponds to the coarse-grained and fine-grained nodes, respectively.*

$$V_0 = V_{para} \cup V_{tbl} \cup V_{img}$$
$$V_1 = V_{sent} \cup V_{row} \cup V_{obj}$$

We denote each vertex set - V_{para}: paragraphs, V_{tbl}: tables, V_{img}: images, V_{sent}: sentences, V_{row}: table rows, V_{obj}: visual objects detected in images. The type map $\tau : V \to \{\texttt{para}, \texttt{tbl}, \texttt{img}, \texttt{sent}, \texttt{row}, \texttt{obj}\}$ refines the vertex set V into the six disjoint categories. The edge set $E \subseteq V \times V$ is the union $E = E_0 \cup E_\downarrow$ where

$$E_0 = \{(u, v) \in V_0^2\}$$
$$E_\downarrow = \{(u, v) \mid u \in V_0, v \in V_1\}$$

E_0 *captures* relationships *between the macro components, while* $E_\downarrow$ *captures the containment of a macro component of its subcomponent.*

The graph $\mathcal{G}$ is constructed in two steps. First, LILaC builds a *component tree* for each component C within $\mathcal{D}$. A component tree is a two-level tree structure with the root representing the component itself and its children representing the subcomponents, which are extracted differently depending on the modality of the component. For a paragraph P, LILaC utilizes a Sentence-aware Transformer (SaT) model to split it into a set of sentences. A table T is parsed to generate a set of table segments. Lastly, a multimodal LLM is used to detect objects within I. LILaC then generates an edge $(C, c) \in E_\downarrow$ for $c \in \mathcal{S}(C)$.

In the next step, LILaC generates the inter-component edges E_0 using both inherent structural relationships and hyperlink-based connections. For every document $D \in \mathcal{D}$, a clique is formed among its components:

$$E_{intra} = \{(C_i, C_j) | C_i \neq C_j, C_i, C_j \in D\} \quad (1)$$

To enable cross-document multihop reasoning, LILaC then follows the link mapping $\mathcal{L}$. For each pair $(C, D) \in \mathcal{L}$, it connects C to every component in the linked document $\mathcal{D}$.

$$E_{inter} = \{(C, C') | (C, D) \in \mathcal{L}, C' \in D\} \quad (2)$$

The inter-component edge set for the top layer is therefore $E_0 = E_{intra} \cup E_{inter}$. Finally, every node $v \in V$ receives an embedding $\mathbf{v} = f(v)$ from a pre-trained multimodal encoder f.

4.2 Late-Interaction-Based Subgraph Retrieval

During the online phase, LILaC retrieves a query-relevant subgraph $\mathcal{G}'$ from the layered component graph $\mathcal{G}$ given a query Q. This retrieval faces two key challenges: (1) Direct identification of an optimal subgraph from all possible candidates is computationally infeasible due to a combinatorial explosion (Hu et al., 2024). In particular, the layered component graph contains numerous edges, making explicit embedding of all edges prohibitively expensive in terms of space and computation. (2) Queries often lack explicit modality instructions, causing ambiguity for multimodal embedders, particularly in complex multihop scenarios (Wei et al., 2024). To address these, we introduce a two-step retrieval strategy: (i) *LLM-driven query decomposition*, which explicitly generates modality-specific subqueries, and (ii) *Late-interaction-guided graph traversal*, a beam-search traversal method dynamically scoring edges based on fine-grained interactions within the low-level nodes.

4.2.1 LLM-driven Query Decomposition

Given a potentially complex query Q, LILaC first leverages an LLM to explicitly decompose Q into simpler modality-specific subqueries. Specifically, we utilize a zero-shot prompting strategy to generate a small set of subqueries:

$$\{q_1, \ldots, q_{k_{sub}}\} = \text{LLM}(Q; \mathit{prompt}_{\text{dec}}) \quad (3)$$

Each subquery is then classified into a modality label $m_j \in \{\texttt{text}, \texttt{table}, \texttt{image}\}$ with a second prompt:

$$m_j = \text{LLM}(q_j; \mathit{prompt}_{\text{mod}}). \quad (4)$$

Using these labels, we obtain modality-specific embeddings $\mathbf{q}_j = f(q_j; m_j)$ for every subquery, while the original query is embedded coarsely as $\mathbf{Q} = f(Q; \varepsilon)$ to seed the initial candidate search. We denote the set of embedded subqueries as $\mathbf{Q}_{\text{sub}} = \{\mathbf{q}_1, \ldots, \mathbf{q}_{k_{\text{sub}}}\}$. Full prompt templates appear in §F.

4.2.2 Late-Interaction-Guided Graph Traversal

At inference time, LILaC searches for a subgraph $\mathcal{G}' \subseteq \mathcal{G}$ that best matches the query. LILaC maintains a beam of size b and iteratively identify a candidate subgraph $\mathcal{G}_t = (V_t, E_t, \lambda, \tau)$ consisting of b edges. Initially, to efficiently narrow the search space from numerous candidate nodes, LILaC identifies a set of top-b top-level nodes V_0 most relevant to the query.

$$V_0 = \arg\max_{C \in V_0}{}^{b} \text{sim}(\mathbf{Q}, \mathbf{C}), \quad E_0 = \{\}. \quad (5)$$

LILaC then initiates iterative traversal of the graph starting from these candidate nodes. In each iteration, LILaC first expands the candidate nodes via one-hop traversal to consider adjacent nodes, dynamically computing query-relevance scores for all edges formed by these expansions. Subsequently, only the top-b scored edges are retained for the next iteration forming subgraph, and their constituent nodes become the new set of candidate nodes, forming $\mathcal{G}_i = (V_i, E_i, \lambda, \tau)$. After the final iteration n_i, LILaC returns the top-n_{ret} nodes from the final subgraph $\mathcal{G}_{n_i}$.

Late Interaction Edge Scoring. As previously discussed, naively calculating edge scores negatively impacts both effectiveness and efficiency.

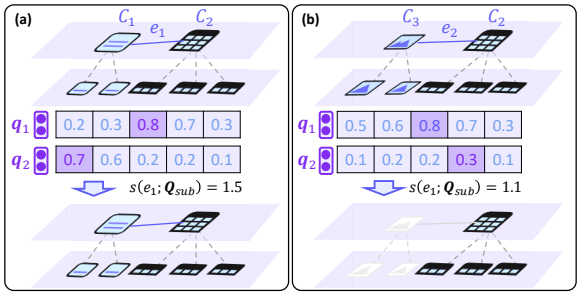


Figure 3: An example case of edge-level late interaction.

Specifically, this is because (1) subqueries, each potentially targeting distinct modalities, must accurately align with the relevant nodes, and (2) embedding all edges within the layered graph is inefficient due to their vast number.

To efficiently address these issues, LILaC employs a *late interaction* strategy, scoring each edge on-the-fly with *fine-grained* evidence. Let an edge be $e = (C_\alpha, C_\beta)$ and $\mathcal{S}_e = \mathcal{S}(C_\alpha) \cup \mathcal{S}(C_\beta)$. LILaC gathers every subcomponent that could provide evidence on either side of the edge in the set $\mathcal{S}_e$.

$$s(e; \mathbf{Q}_{sub}) = \sum_{\mathbf{q} \in \mathbf{Q}_{sub}} \max_{c \in \mathcal{S}_e} \mathrm{sim}\big(f(c), \mathbf{q}\big). \quad (6)$$

The inner max selects, for each sub-query $\mathbf{q}$, the single most relevant sub-component $\mathbf{c}$ incident to the edge, while the outer sum ensures every subquery contributes exactly once. Figure 3 shows two example cases of late interaction scoring. This scoring approach is designed to reflect practical scenarios where each subquery specifically targets fine-grained details located within particular subcomponents. By aggregating the maximum similarity scores across these detailed elements, rather than relying solely on coarse component embeddings, LILaC effectively prioritizes precise, subcomponent-level matches. This strategy enhances retrieval accuracy by focusing directly on relevant information, reducing the noise introduced by broader, less relevant contexts.

We introduce two special cases of edge scoring: *(i) Isolated nodes.* If a component C has no explicit neighbor, we introduce a dummy edge (C, ε) so that C can still be considered. *(ii) One-sided matches.* If an edge score $s(e; Q)$ equals the best single-node score of one endpoint, we return only that node to avoid including irrelevant neighbors. Refer to Figure 3 (b) for a specific example.

5 Experiments

5.1 Experimental Setups

Datasets & Evaluation Metrics. We evaluate on total five benchmarks. Three are VisRAG-extended open-domain VQA datasets—MP-DocVQA (Tito et al., 2023) (industrial documents), SlideVQA (Tanaka et al., 2023)(presentation slides with multi-hop queries), and InfoVQA (Mathew et al., 2022) (infographics). For a realistic webpage retrieval setting, we extend multimodal QA benchmarks (MultimodalQA (Talmor et al., 2021), MMCoQA (Li et al., 2022b)) using M3DocRAG's methodology (Cho et al., 2024). Specifically, we reconstruct webpages from URLs annotated in each component label. MultimodalQA comprises 3,235 webpages, each averaging approximately 37 components, corresponding to about 12 PDF pages. MMCoQA comprises 453 webpages, each averaging approximately 32 components, 11 PDF pages.

Following VisRAG, we evaluate retrieval using Mean Reciprocal Rank at 10 (MRR@10). Additionally, we include Recall@3 to assess whether the retrieval component successfully captures relevant information within the top three components, aligning with VisRAG's experimental design that inputs three components to the generation model. Further details are explained in § E.2.

Compared Methods. We employ two SOTA methods of VisRAG approaches - VisRAG, which directly encodes document images via VLMs (Yu et al., 2024), and ColPali, which employs late-interaction multi-vector embeddings from document images (Faysse et al., 2024). We additionally compare with NV-Embed-v2, a SOTA TextRAG method reported by VisRAG. It utilizes a 7.85B model for embedding textualized components.

Applied Multimodal Embedding Models. We use three multimodal embedders: MM-Embed (Lin et al., 2024), UniME (Gu et al., 2025) and mmE5 (Chen et al., 2025). Details about the embedding models can be further found in § D.

5.2 Retrieval Accuracy Comparison

We evaluated retrieval accuracies using Recall@3 (R@3) and MRR@10 across five benchmarks. Table 1 summarizes the retrieval performance of LILaC and competing methods. Our results indicate that LILaC achieves state-of-the-art (SOTA) performance on four of the five benchmarks, specifically on MP-DocVQA, SlideVQA, MultimodalQA, and MMCoQA. Notably, LILaC outperforms the previous VisRAG SOTA models, VisRAG-Ret and ColPali, by substantial margins of 12.39% and 9.85% in R@3, and 14.45% and 10.49% in MRR@10, on average, respectively. These performance gains

Algorithm	Embedder Type	MP-DocVQA		SlideVQA		InfoVQA		MultimodalQA		MMCoQA	
		R@3	MRR@10	R@3	MRR@10	R@3	MRR@10	R@3	MRR@10	R@3	MRR@10
NV-Embed-v2	Text	67.85	61.91	88.49	79.55	86.21	80.86	60.19	67.86	46.16	41.45
VisRAG-Ret	Image	83.25	75.55	91.55	84.30	**92.76**	**86.22**	50.08	55.08	27.63	23.75
ColPali	Image	80.71	74.86	89.39	81.55	88.30	82.76	58.73	65.05	36.24	32.33
LILaC (w/ mmE5)	Multimodal	61.25	55.30	77.52	68.80	70.33	65.74	54.79	59.02	48.88	40.30
LILaC (w/ UniME)	Multimodal	77.83	71.42	84.35	77.93	78.83	72.74	58.52	61.44	49.63	42.97
LILaC (w/ MM-Embed)	Multimodal	**83.59**	**78.75**	**92.81**	**84.43**	86.91	82.63	**69.07**	**75.28**	**55.80**	**50.77**

Table 1: Retrieval accuracy (Recall@3 (*R@3*) and *MRR@10*) of LILaC and its competitors on five benchmarks. The best score in each column is in **bold**. The in-domain fine-tuned settings are colored in orange.

Algorithm	MLLM	MP-DocVQA		SlideVQA		InfoVQA		MultimodalQA		MMCoQA	
		EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
NV-Embed-v2	Qwen2.5-VL 7B	56.51	63.16	53.77	64.41	60.72	63.40	37.23	43.85	28.05	34.67
VisRAG-Ret	MiniCPM V2.6	54.31	68.86	43.88	62.37	50.83	57.55	28.18	34.01	21.51	27.87
VisRAG-Ret	Qwen2.5-VL 7B	65.34	72.24	55.03	66.13	**60.16**	**61.93**	22.24	25.55	16.69	20.90
ColPali	Qwen2.5-VL 7B	64.46	71.16	53.77	64.54	58.07	60.38	23.59	27.37	18.07	22.30
LILaC (w/ mmE5)	Qwen2.5-VL 7B	52.96	59.53	50.89	59.07	49.44	51.81	40.72	47.46	33.90	40.38
LILaC (w/ UniME)	Qwen2.5-VL 7B	62.43	69.40	53.05	62.89	52.78	54.47	43.42	49.72	33.39	40.12
LILaC (w/ MM-Embed)	Qwen2.5-VL 7B	**65.48**	**72.42**	**55.57**	**66.32**	58.07	60.04	**44.57**	**51.97**	**36.31**	**43.22**

Table 2: End-to-end accuracy (EM and F1) of LILaC and its competitors for the 5 benchmarks. The best score in each column is in **bold**. Generation results corresponding to in-domain fine-tuned settings are colored in orange.

are especially prominent on datasets that inherently require fine-grained and multihop reasoning (MultimodalQA and MMCoQA), where the relative improvements in average Recall@3 reached 60.68% and 31.49%, and MRR@10 improved by 59.90% and 45.92%, respectively.

Our analysis highlights three key findings: (i) TextRAG of NV-Embed-v2, consistently shows the lowest retrieval accuracy on visually-dependent VQA datasets that include plots and charts, highlighting inherent limitations in handling visual modalities. (ii) VisRAG methods notably struggle in webpage retrieval settings (MultimodalQA, MMCoQA), underperforming even when compared to the text-based NV-Embed-v2. Specifically, the stronger VisRAG model, ColPali, showed accuracy drops against NV-Embed-v2, with reductions of 10.70% in Recall@3 and 20.96% in MRR@10. (iii) Finally, LILaC underperformed VisRAG methods on InfoVQA, achieving R@3 and MRR@10 scores lower by 6.3% and 4.16% than VisRAG-Ret, respectively. Our subsequent analysis attributes this specific gap primarily to suboptimal subcomponent detection within image components in InfoVQA, leading to ineffective late interaction.

5.3 End-to-end Accuracy Comparison

We conducted end-to-end question answering (QA) experiments to analyze the impact of retrieval accuracy on downstream QA performance. The retrieved results were directly input into a multimodal LLM generator for answer generation, primarily using the Qwen2.5-VL 7B model (Yang et al., 2024). We limited the number of retrieved units fed into the generator to 3, consistent with the experimental setup of VisRAG. We additionally provide the results from MiniCPM V2.6 for comprehensive comparison, following the original VisRAG pipeline. Applied prompts are detailed in § F.

Table 2 shows that LILaC achieves SOTA average end-to-end accuracy, with average EM and F1 scores of 52.00 and 58.79, respectively. This represents substantial improvements of 17.40% and 18.47% compared to the previously best-performing VisRAG setup, VisRAG with Qwen2.5-VL, which scored 44.29 (EM) and 49.62 (F1). Overall, the end-to-end QA accuracy trends closely align with retrieval accuracy. Interestingly, despite LILaC (w/ mmE5) having approximately 8.97% lower retrieval accuracy (R@3) compared to NV-Embed-v2, its EM score surpasses NV-Embed-v2 by 19.71This divergence highlights the significant information loss inherent to TextRAG methods, which convert visual content entirely into text, underscoring the importance of preserving visual modalities for effective QA.

5.4 Ablation Study

We performed an ablation study to assess the individual contributions of each key component in our framework to retrieval accuracy. Specifically, we evaluated two simplified variants of LILaC across all three multimodal embedding models. The first variant, *Top-layer kNN*, directly applies a

Embedder Model	Variant	MP-DocVQA		SlideVQA		InfoVQA		MultimodalQA		MMCoQA	
		R@3	MRR@10	R@3	MRR@10	R@3	MRR@10	R@3	MRR@10	R@3	MRR@10
mmE5	Component kNN	48.90	43.97	75.91	68.13	65.60	58.55	42.99	46.92	41.22	34.51
	Layered graph search	60.81	55.02	74.14	67.58	69.78	64.12	45.15	51.12	44.18	36.62
	LILaC	**61.25**	**55.35**	**76.80**	**68.99**	**70.19**	**65.65**	**54.78**	**59.32**	**48.54**	**40.22**
UniME	Component kNN	52.12	45.31	81.47	71.22	83.57	77.07	47.68	49.06	45.78	38.41
	Layered Graph Search	77.83	71.27	83.45	75.70	78.41	72.28	52.18	54.01	47.11	39.85
	LILaC	**77.83**	**71.39**	**84.35**	**77.93**	**78.83**	**72.72**	**58.43**	**61.32**	**49.45**	**42.91**
MM-Embed	Component kNN	75.80	69.09	92.80	82.19	90.39	83.71	61.10	67.35	47.94	43.75
	Layered Graph Search	82.23	77.75	92.27	83.20	84.12	80.08	63.19	69.91	50.18	45.59
	LILaC	**83.59**	**78.75**	**92.81**	**84.43**	**86.91**	**82.63**	**69.07**	**75.28**	**55.80**	**50.77**

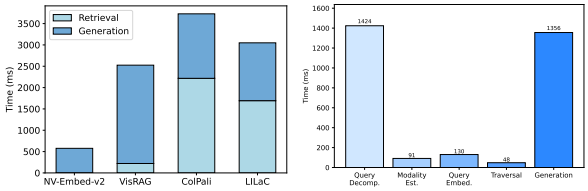
Table 3: Ablation study analyzing retrieval accuracy (Recall@3 and MRR@10) of different LILaC variants. Best scores per embedder and dataset are highlighted in bold.

k-nearest neighbor search on individual top-layer components without leveraging finer-grained subcomponents. The second variant, *Layered Graph Search*, incorporates a two-stage retrieval approach on the layered graph: it first selects the top b-nearest neighbor components at the coarse level, and then reranks these components by considering subcomponent-level relevance scores.

As shown in Table 3, employing the layered graph structure leads to notable average improvements - 7.33% in R@3 and 10.13% in MRR@10—over the simple component-only baseline. Integrating query decomposition with the late interaction mechanism yields further incremental gains of 3.19% in R@3 and 4.7% in MRR@10. While these improvements seem modest, closer inspection reveals significant benefits in datasets requiring complex multihop reasoning, particularly MultimodalQA and MMCoQA. Specifically, incorporating query decomposition and late interaction improves R@3 by an average of 7.40% and MRR@10 by 10.70% for these two datasets. Overall, LILaC is demonstrated to be a general method, as evidenced by its consistent performance improvements across diverse multimodal datasets (with an exception of InfoVQA - refer to § 5.2) and embedding models. This robust trend underscores LILaC's ability to universally enhance retrieval performance across a variety of multimodal embedding scenarios.

5.5 Algorithm Execution Time

Figure 4 (a) shows the average retrieval and generation times for each algorithm. LILaC is approximately 20.76% slower than VisRAG, yet 18.24% faster than ColPali. Despite employing a unigranular retrieval approach, ColPali's runtime remained slower due to its inherent complexity from multivector embedding methods. Notably, both VisRAG methods had longer generation times compared to ours—VisRAG required 1.70×, and ColPali



(a) Average algorithm runtime (b) Breakdown of LILaC

Figure 4: (a) Comparison of average algorithm execution times across different methods, and (b) detailed runtime breakdown of LILaC.

1.15× times our average generation runtime, primarily because their pixel-heavy image inputs increased MLLM inference times.

Figure 4 (b) presents the detailed runtime breakdown for LILaC, showing a total average runtime of 3,047 ms. Remarkably, the late-interaction-based subgraph retrieval step accounts for only about 48 ms (approximately 1.5% of the total runtime). The major performance bottleneck lies in the query decomposition phase, averaging 1,423 ms. Since this step relies on advanced reasoning with the computationally heavy Qwen2.5 72B model, future improvements in runtime efficiency could be realized by utilizing lighter models, thus balancing speed and retrieval accuracy more effectively.

6 Conclusion

We presented LILaC, a multimodal retrieval framework designed to address the limitations of existing methods by incorporating layered component graph and late-interaction-based subgraph retrieval. Our layered graph construction explicitly captures semantic relationships among multimodal components, facilitating effective multihop reasoning. The late-interaction retrieval method dynamically evaluates fine-grained component relevance, significantly enhancing retrieval accuracy, yet efficient. Extensive experiments confirm that LILaC consistently outperforms state-of-the-art approaches across four out of five benchmarks, also demonstrating its broad applicability and effectiveness in open-domain multimodal retrieval.

7 Limitations

Our current approach focuses on effectively harmonizing pre-trained multimodal models to achieve enhanced retrieval performance without additional fine-tuning. Consequently, the accuracy of our retrieval method significantly depends on the quality of subcomponent extraction, especially within image and table modalities. As demonstrated in our empirical analysis (e.g., with the `InfoVQA` dataset), inaccuracies during subcomponent extraction can negatively affect retrieval quality. Lastly, although our retrieval accuracy surpasses existing methods, there remains substantial room for improvement in end-to-end generation tasks.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*.

Yingshan Chang, Mridu Narang, Hisami Suzuki, Guihong Cao, Jianfeng Gao, and Yonatan Bisk. 2022. Webqa: Multihop and multimodal qa. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16495–16504.

Haonan Chen, Liang Wang, Nan Yang, Yutao Zhu, Ziliang Zhao, Furu Wei, and Zhicheng Dou. 2025. mme5: Improving multimodal multilingual embeddings via high-quality synthetic data. *arXiv preprint arXiv:2502.08468*.

Tong Chen, Hongwei Wang, Sihao Chen, Wenhao Yu, Kaixin Ma, Xinran Zhao, Hongming Zhang, and Dong Yu. 2024. Dense x retrieval: What retrieval granularity should we use? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 15159–15177.

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*.

Manuel Faysse, Hugues Sibille, Tony Wu, Bilel Omrani, Gautier Viaud, Céline Hudelot, and Pierre Colombo. 2024. Colpali: Efficient document retrieval with vision language models. In *The Thirteenth International Conference on Learning Representations*.

Tiancheng Gu, Kaicheng Yang, Ziyong Feng, Xingjun Wang, Yanzhao Zhang, Dingkun Long, Yingda Chen, Weidong Cai, and Jiankang Deng. 2025. Breaking the modality barrier: Universal embedding learning with multimodal llms. *arXiv preprint arXiv:2504.17432*.

Jonathan Herzig, Thomas Müller, Syrine Krichene, and Julian Martin Eisenschlos. 2021. Open domain question answering over tables via dense retrieval. *arXiv preprint arXiv:2103.12011*.

Yuntong Hu, Zhihan Lei, Zheng Zhang, Bo Pan, Chen Ling, and Liang Zhao. 2024. Grag: Graph retrieval-augmented generation. *arXiv preprint arXiv:2405.16506*.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Ziyan Jiang, Rui Meng, Xinyi Yang, Semih Yavuz, Yingbo Zhou, and Wenhu Chen. 2024. Vlm2vec: Training vision-language models for massive multimodal embedding tasks. *arXiv preprint arXiv:2410.05160*.

Dongxu Li, Junnan Li, and Steven Hoi. 2023. Blip-diffusion: Pre-trained subject representation for controllable text-to-image generation and editing. *Advances in Neural Information Processing Systems*, 36:30146–30166.

Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. 2022a. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International conference on machine learning*, pages 12888–12900. PMLR.

Yongqi Li, Wenjie Li, and Liqiang Nie. 2022b. Mmcoqa: Conversational question answering over text, tables, and images. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4220–4231.

Sheng-Chieh Lin, Chankyu Lee, Mohammad Shoeybi, Jimmy Lin, Bryan Catanzaro, and Wei Ping. 2024. Mm-embed: Universal multimodal retrieval with multimodal llms. *arXiv preprint arXiv:2411.02571*.

Haohao Luo, Ying Shen, and Yang Deng. 2023. Unifying text, tables, and images for multimodal question answering. Association for Computational Linguistics.

Minesh Mathew, Viraj Bagal, Rubèn Tito, Dimosthenis Karatzas, Ernest Valveny, and CV Jawahar. 2022. Infographicvqa. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1697–1706.

Lang Mei, Siyu Mo, Zhihan Yang, and Chong Chen. 2025. A survey of multimodal retrieval-augmented generation. *arXiv preprint arXiv:2504.08748*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, and 1 others. 2021. Learning transferable visual models from natural language supervision. In *International*

conference on machine learning, pages 8748–8763. PmLR.

Monica Riedler and Stefan Langer. 2024. Beyond text: Optimizing rag with multimodal inputs for industrial applications. *arXiv preprint arXiv:2410.21943*.

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*.

Alon Talmor, Ori Yoran, Amnon Catav, Dan Lahav, Yizhong Wang, Akari Asai, Gabriel Ilharco, Hannaneh Hajishirzi, and Jonathan Berant. 2021. Multimodalqa: Complex question answering over text, tables and images. *arXiv preprint arXiv:2104.06039*.

Ryota Tanaka, Kyosuke Nishida, Kosuke Nishida, Taku Hasegawa, Itsumi Saito, and Kuniko Saito. 2023. Slidevqa: A dataset for document visual question answering on multiple images. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 13636–13645.

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

Cong Wei, Yang Chen, Haonan Chen, Hexiang Hu, Ge Zhang, Jie Fu, Alan Ritter, and Wenhu Chen. 2024. Uniir: Training and benchmarking universal multimodal information retrievers. In *European Conference on Computer Vision*, pages 387–404. Springer.

Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. 2024. Corrective retrieval augmented generation.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Qian Yang, Qian Chen, Wen Wang, Baotian Hu, and Min Zhang. 2023. Enhancing multi-modal multi-hop question answering via structured knowledge and unified retrieval-generation. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 5223–5234.

Bowen Yu, Cheng Fu, Haiyang Yu, Fei Huang, and Yongbin Li. 2023a. Unified language representation for question answering over text, tables, and images. *arXiv preprint arXiv:2306.16762*.

Shi Yu, Chaoyue Tang, Bokai Xu, Junbo Cui, Junhao Ran, Yukun Yan, Zhenghao Liu, Shuo Wang, Xu Han, Zhiyuan Liu, and 1 others. 2024. Visrag: Vision-based retrieval-augmented generation on multi-modality documents. *arXiv preprint arXiv:2410.10594*.

Zichun Yu, Chenyan Xiong, Shi Yu, and Zhiyuan Liu. 2023b. Augmentation-adapted retriever improves generalization of language models as generic plug-in. *arXiv preprint arXiv:2305.17331*.

Zijie Zhong, Hanwen Liu, Xiaoya Cui, Xiaofan Zhang, and Zengchang Qin. 2024. Mix-of-granularity: Optimize the chunking granularity for retrieval-augmented generation. *arXiv preprint arXiv:2406.00456*.

Appendix

A Software and Data Licenses

The licenses for the software and datasets used in this paper as follows:

- VisRAG-Ret: Apache-2.0
- ColPali: PaliGemma License, MIT License
- MiniCPM-v2.6: Apache-2.0
- Qwen2.5-VL 7B: Apache-2.0
- Qwen2.5 72B: Qwen
- MM-Embed: CC-BY-NC-4.0
- NV-Embed-v2: CC-BY-NC-4.0
- UniME: MIT License
- mmE5: MIT License

All software and datasets were used strictly for research purposes and were not utilized in any non-research contexts, particularly for commercial applications.

B AI Assistants

We implemented our code efficiently using ChatGPT-o3 (Jaech et al., 2024), enabling rapid debugging and effective error resolution. Additionally, we revised our paper using ChatGPT-4.5, which helped us enhance sentence clarity and readability through iterative rephrasing.

C Reproducibility Statement

VisRAG-Ret was reproduced using the official code available at VisRAG official github. ColPali and NV-Embed-v2 were implemented applying their official model cards introduced in ColPali huggingface and NV-Embed-v2 huggingface, respectively. The source code, data, and other artifacts for LILaC have been made available at our anonymous github repository.

D Model Details

- Qwen2.5-VL 7B: 7B parameters
- MiniCPM-v2.6: 8.1B parameters

LLMs:

- Qwen2.5 72B: 72B parameters

Text embedders

- NV-Embed-v2:

Cross-modal embedders:

- ColPali: 3B parameters
- VisRAG-Ret: 3.43B parameters

Multimodal embedders:

- MM-Embed: 8.18B parameters
- UniME: 7.57B parameters
- mmE5: 10.6B parameters

MM-Embed is fine-tuned via modality-aware hard negative mining (Lin et al., 2024). UniME is enhanced with textual discriminative knowledge distillation and instruction-tuned hard negatives (Gu et al., 2025). mmE5 leverages synthetic multilingual data for robust cross-modal alignment (Chen et al., 2025).

Multimodal LLMs:

E Experiment Supplementaries

E.1 Hardware and Software Settings

All our experiments were conducted on a system with an Intel Xeon Gold 6230 GPU @ 2.10GHz, 1.5TB of RAM, and four NVIDIA RTX A6000 GPUs.

E.2 Implementation Details

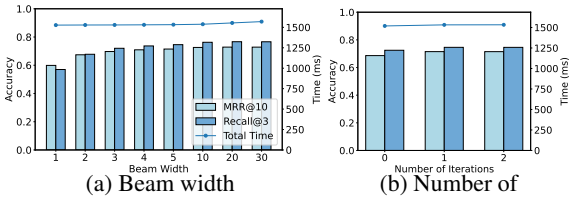
We set the default hyperparameters for all experiments as beam width $b = 30$ and number of iterations $n_i = 1$. Additionally, for the ablation study that exclusively uses the layered graph structure without late interaction, we also maintained an identical beam width ($b = 30$) to ensure a fair comparison.

All experiments were conducted with 'temperature = 0' and 'do_sample = False'. To further ensure fair comparison, we aligned the ratio of components between the VisRAG methods and our approach to approximately 1:3, as justified by the empirical observation that a typical screenshot in our datasets encompasses roughly three distinct multimodal components. Specifically, the MultimodalQA dataset contains 39,093 screenshots and 122,521 components, and the MMCoQA dataset comprises 5,175 screenshots and 14,493 components, both yielding a component-to-screenshot ratio close to 3:1.

E.3 Benchmark Details

MP-DocVQA: MP-DocVQA is a multimodal visual question answering benchmark designed for industrial documents. It includes challenging questions that require extracting and reasoning over textual and visual information such as tables, figures, and charts found in documents. The development set contains 591 questions sourced from a corpus of 741 multimodal document pages

SlideVQA: SlideVQA focuses on extracting information from presentation slides and often re-



(a) Beam width (b) Number of iterations

Figure 5: Change in retrieval accuracy with varying parameter values.

quires multihop reasoning across multiple slides. It emphasizes the capability to handle diverse layouts and structured textual information commonly found in presentations. The SlideVQA development set comprises 556 questions, with the corpus containing 1,284 slide pages

InfoVQA: InfoVQA targets visual question answering on infographics, which blend images, charts, and textual descriptions. This dataset presents complex multimodal reasoning tasks where models must interpret visual elements combined with succinct textual explanations. Its development set includes 718 questions drawn from a corpus of 459 infographic pages

MultimodalQA: MultimodalQA, referred to the extended version of MultimodalQA introduced in M3DocRAG (Cho et al., 2024), evaluates open-domain multimodal document understanding and reasoning. The dataset covers a wide variety of document types, including texts, images, and tables, requiring complex multihop reasoning across multiple documents. Its evaluation set comprises 2,441 questions from over 3,368 PDF documents totaling approximately 41,005 pages

MMCoQA: MMCoQA is a conversational multimodal question-answering dataset aimed at testing a system's ability to handle multimodal information across multiple turns in a conversational context. It involves coherent, multi-turn question sequences requiring integration of information from text, images, and tables. The dataset includes 5,753 questions organized into 1,179 conversational dialogues. Its corpus consists of 218,285 textual passages, 10,042 tables, and 57,058 images

E.4 Parameter Sensitivity

We explored the impact of varying the beam width $b(\in 1, 2, 3, 4, 5, 10, 20, 30)$ on the retrieval accuracies. As depicted in Figure 5 (a), retrieval accuracy increased monotonically with larger beam widths, showing a significant improvement of 34.6% in R@3 when expanding from the minimum of 1 to 30. This trend highlights the benefit of wider beam searches, enabling more comprehensive and accurate graph traversal. Interestingly, despite these substantial accuracy gains, the overall execution time increased only marginally (2.8%), indicating that graph traversal itself does not constitute the main computational bottleneck.

Figure 5 (b) presents retrieval accuracy as a function of iteration count n_i, varied from 0 to 2. We observed a modest yet meaningful 2.93% improvement in R@3 when transitioning from zero to one iteration. This accuracy gain primarily results from enabling multihop reasoning, which is inherently unavailable at $n_i = 0$. While the overall increase might appear limited, it is particularly relevant to datasets explicitly requiring complex multihop reasoning, such as `MultimodalQA` and `MMCoQA`.

E.5 Parameter Sensitivity: Detailed Results

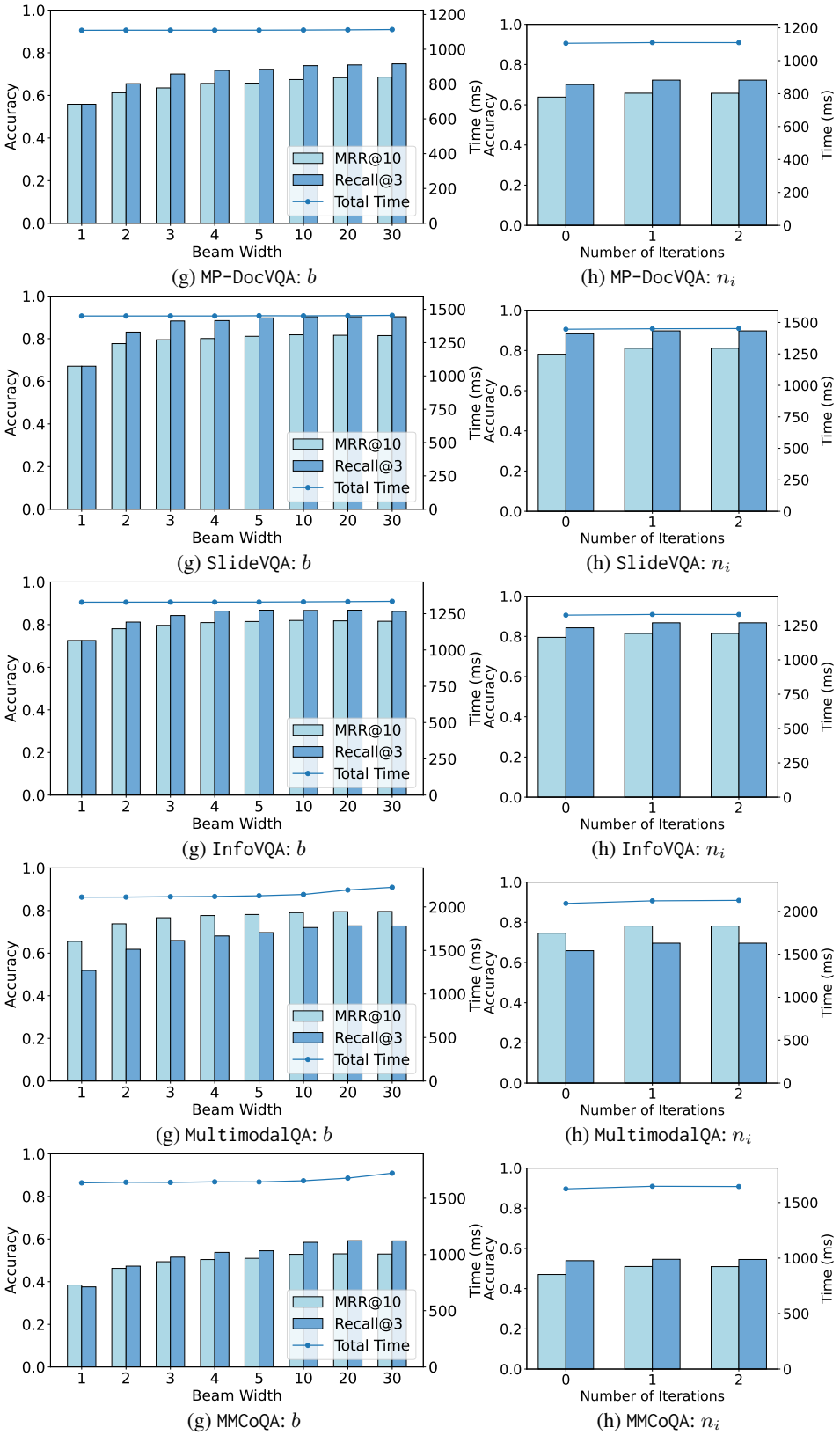
In this section we present the detailed analyses on how varying key hyperparameters—specifically, beam width b and the number of iterations n_i, - affect the performance across different datasets (`MP-DocVQA`, `SlideVQA`, `InfoVQA`, `MultimodalQA`, and `MMCoQA`). We provided comprehensive plots illustrating the sensitivity and robustness of our method concerning these parameters in Figure 6.

E.6 Algorithm Execution Runtime: Detailed Results

We conducted an in-depth examination of runtime efficiency. Specifically, we compared the overall execution time of our proposed method, `LILaC`, against other baseline algorithms across all datasets. We further broke down `LILaC`'s runtime into individual components (such as retrieval, reranking, and LLM refinement) to clearly identify performance bottlenecks and highlight the efficiency of different pipeline stages. Detailed results are shown in Figure 7.

F Prompt Templates

We present detailed examples of the specific prompt templates used in our experiments. These prompts correspond to three key tasks: Object Detection, Query Decomposition, Modality Selection and Answer Generation. For each task, we provide clear instructions, expected input-output formats, and task-specific heuristics.



(g) MP-DocVQA: b (h) MP-DocVQA: n_i

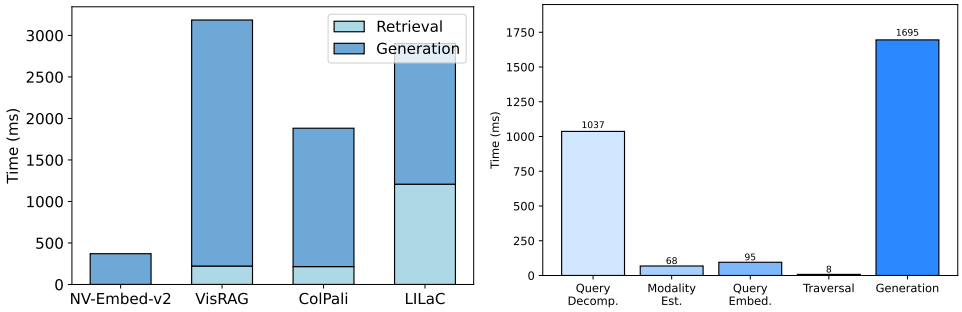
(g) SlideVQA: b (h) SlideVQA: n_i

(g) InfoVQA: b (h) InfoVQA: n_i

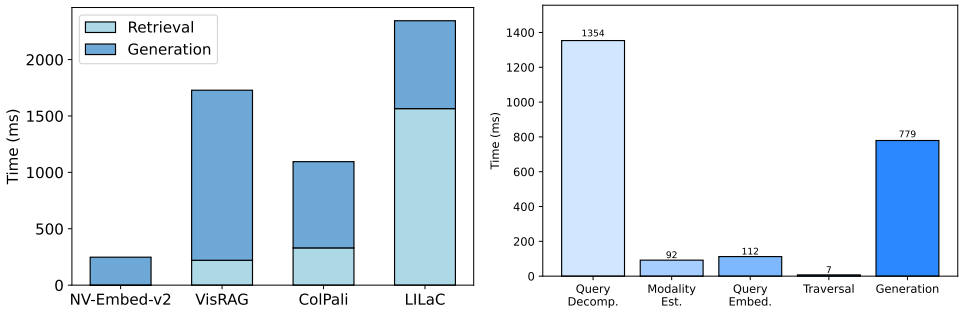
(g) MultimodalQA: b (h) MultimodalQA: n_i

(g) MMCoQA: b (h) MMCoQA: n_i

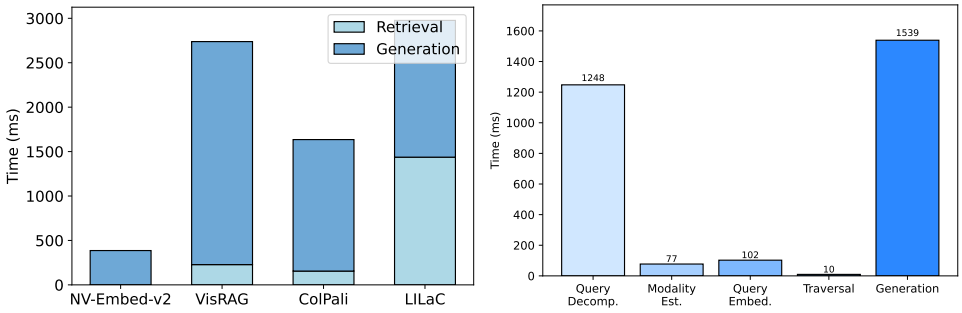
Figure 6: Parameter-sensitivity analysis for each dataset: effect of beam width b (left) and number of iterations n_i (right).



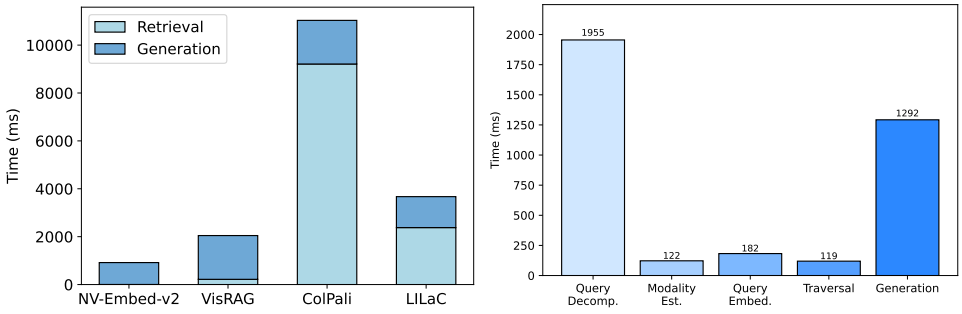
(a) MP-DocVQA runtime comparison (b) LILaC's runtime breakdown on MP-DocVQA



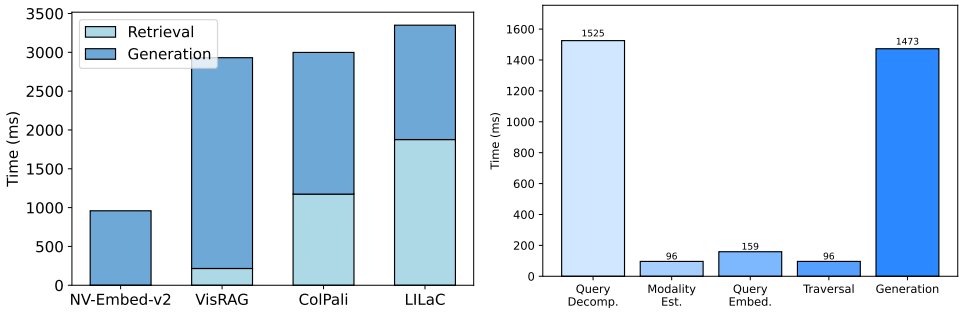
(c) SlideVQA runtime comparison (d) LILaC's runtime breakdown on SlideVQA



(e) InfoVQA runtime comparison (f) LILaC's runtime breakdown on InfoVQA



(g) MultimodalQA runtime comparison (h) LILaC's runtime breakdown on MultimodalQA



(i) MMCoQA runtime comparison (j) LILaC's runtime breakdown on MMCoQA

Figure 7: Comparison of algorithm execution time (i.e., runtime) for each algorithm per dataset (left) and LILaC's runtime breakdown per dataset (right) of LILaC (right).

Object Detection

Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.

Instruction:
Detect all objects in the image and return **ONLY** a JSON list of {class, bbox_2d:[x1,y1,x2,y2]}. Do **NOT** include markdown or extra text.

Input:
Image: `{image}`
Output:

Query Decomposition

Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.

Instruction:
You are a retrieval-oriented **query decomposer**.

Goal – Produce the smallest set (1 – 5) of **component-targeting sub-queries**.
Each sub-query must describe **one retrievable component** (sentence, paragraph, table row, figure, etc.) whose embedding should be matched.
Together, the sub-queries must supply all the information needed to answer the original question.

Guidelines
1. **Entity & noun-phrase coverage** Every noun phrase and named entity that appears in the original question must appear **at least once across the entire set** of sub-queries (you may distribute them). Keep each phrase exactly as written.
2. **One-component rule** A sub-query should reference only the facts expected to co-occur **within the same component**. If two facts will likely be in different components, put them in different sub-queries.
3. **No unnecessary splitting** If the whole answer can be found in a single component, return only one sub-query.
4. **De-contextualize** Rewrite pronouns and implicit references so every sub-query is understandable on its own.
5. **Keyword distribution** Spread constraints logically (e.g., one sub-query for "light rail completion date", another for "city with a large arched bridge from the 1997 Australia rugby-union test match").
6. **Remove redundancy** Merge duplicate or paraphrased sub-queries before you output.
7. **Ordering for dependencies** If the answer to one sub-query is needed for another, place the prerequisite first.
8. **Output format** Return **only** a JSON array of strings — no keys, explanations, or extra text.

Input:
Question: `{question}`
Output:

Modality Selection

Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.

Instruction:
You are a modality selector for multimodal QA.

Task
Given the single sub-question below, choose the **one** modality that is most appropriate for obtaining its answer.

Allowed modalities
• text – unstructured prose (paragraphs, sentences, propositions)
• table – structured rows/columns (spreadsheets, stats tables, infoboxes)
• image – visual information (photos, posters, logos, charts)

Heuristics
1. Numeric totals, percentages, year-by-year figures → table
2. Visual appearance, colours, logos, "what does … look like" → image
3. Definitions, roles, biographies, causal explanations, quotes → text
4. If two modalities could work, pick the one that will yield the answer **fastest**.

Output format
Return **only** the modality label on a single line – exactly `text`, `table`, or `image`.
No JSON, no additional text.

Input:
Subquery: `{subquery}`
Output:

Answer Generation Prompt

Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.

Instruction:
Using the `f_answers()` API, return a list of answers to the question based on *retrieved webpage components*.
A retrieved component can be a passage, a table, or an image.
Strictly follow the format of the example below and keep the answer **short**.
For *yes/no* questions, answer with either `f_answers(["yes"])` or `f_answers(["no"])` only.

Example
Retrieved components

[Passage] **Title: South Asia**
The current territories of Afghanistan, Bangladesh, Bhutan, Maldives, Nepal, India, Pakistan, and Sri Lanka form South Asia. The South Asian Association for Regional Cooperation (SAARC) is an economic cooperation organisation in the region which was established in 1985 and includes all eight nations comprising South Asia.

[Passage] **Title: UK Joint Expeditionary Force**
The UK Joint Expeditionary Force (JEF) is a United Kingdom-led expeditionary force which may consist of, as necessary, Denmark, Finland, Estonia, Latvia, Lithuania, the Netherlands, Sweden and Norway. It is distinct from the similarly named Franco-British Combined Joint Expeditionary Force.

[Table] **Title: Lithuanian Armed Forces — Current operations**

Deployment Organization Operation Personnel

Somalia EU Operation Atalanta 15
Mali EU EUTM Mali 2
Afghanistan NATO Operation Resolute Support 29
Libya EU EU Navfor Med 3
Mali UN MINUSMA 39
Iraq CJTF Operation Inherent Resolve 6
Central African Republic EU EUFOR RCA 1
Kosovo NATO KFOR 1
Ukraine Training mission 40
Question
Among the Lithuanian Armed Forces' current operations, which deployment involves fewer personnel: *Kosovo*, or the deployment in the nation that, along with six others, constitutes the sub-continent of South Asia?
Explanation
Afghanistan is listed as part of South Asia. The table shows 29 personnel in Afghanistan and only 1 in Kosovo; therefore `f_answers(["Kosovo"])`.

Input:
Using the images and texts given, answer the question below in a single word or phrase.

Question: `{question}`
Answer: