DATABASE LAB @POSTECH

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

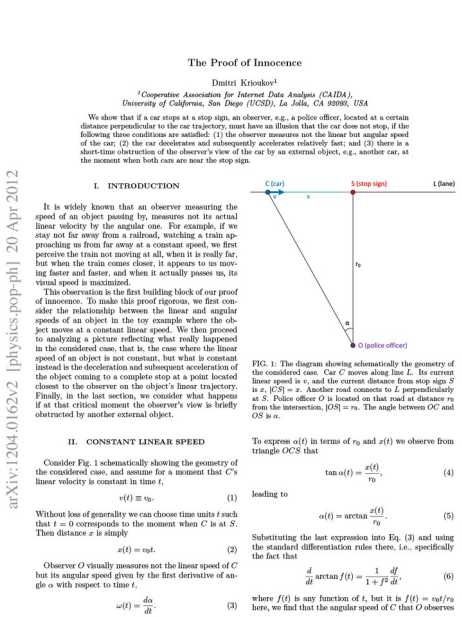
Joohyung Yun, Doyup Lee, Wook-Shin Han*

30th ANNIVERSARY EMNLP 2025
Suzhou, China | 中国苏州
November 4-9 | 11月4日–9日

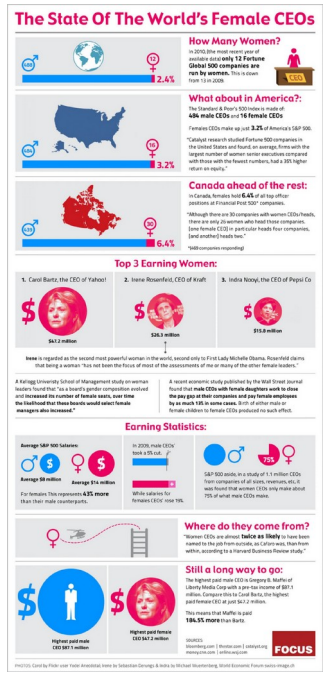
Multimodal Documents

- Real-world documents are often multimodal.
  - Examples of such multimodal documents include webpages, PDFs, presentation slides, etc.
- We refer to the elements that make up a document as *multimodal components*.
  - Correctly resolving the relationships between the components is a key factor in performing question-answering tasks.

An example of a multimodal webpage



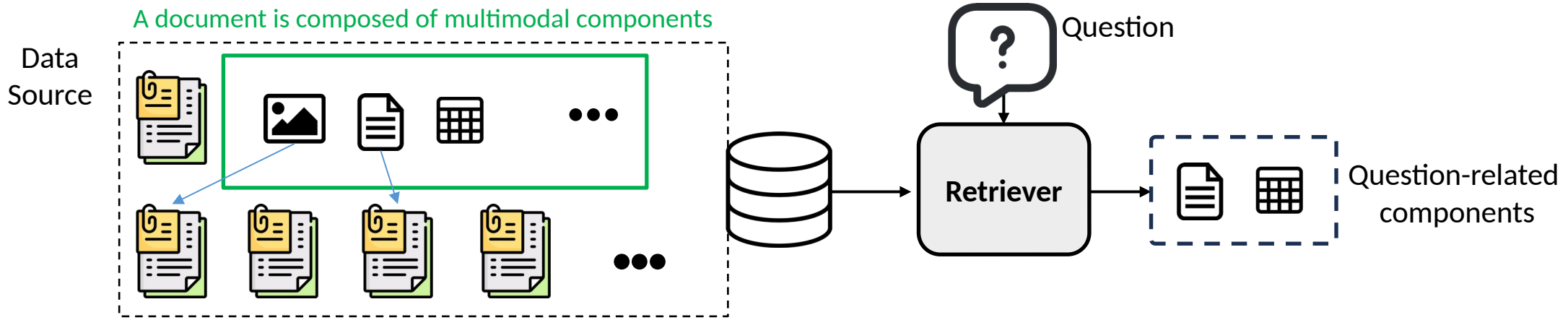
An example of a PDF file



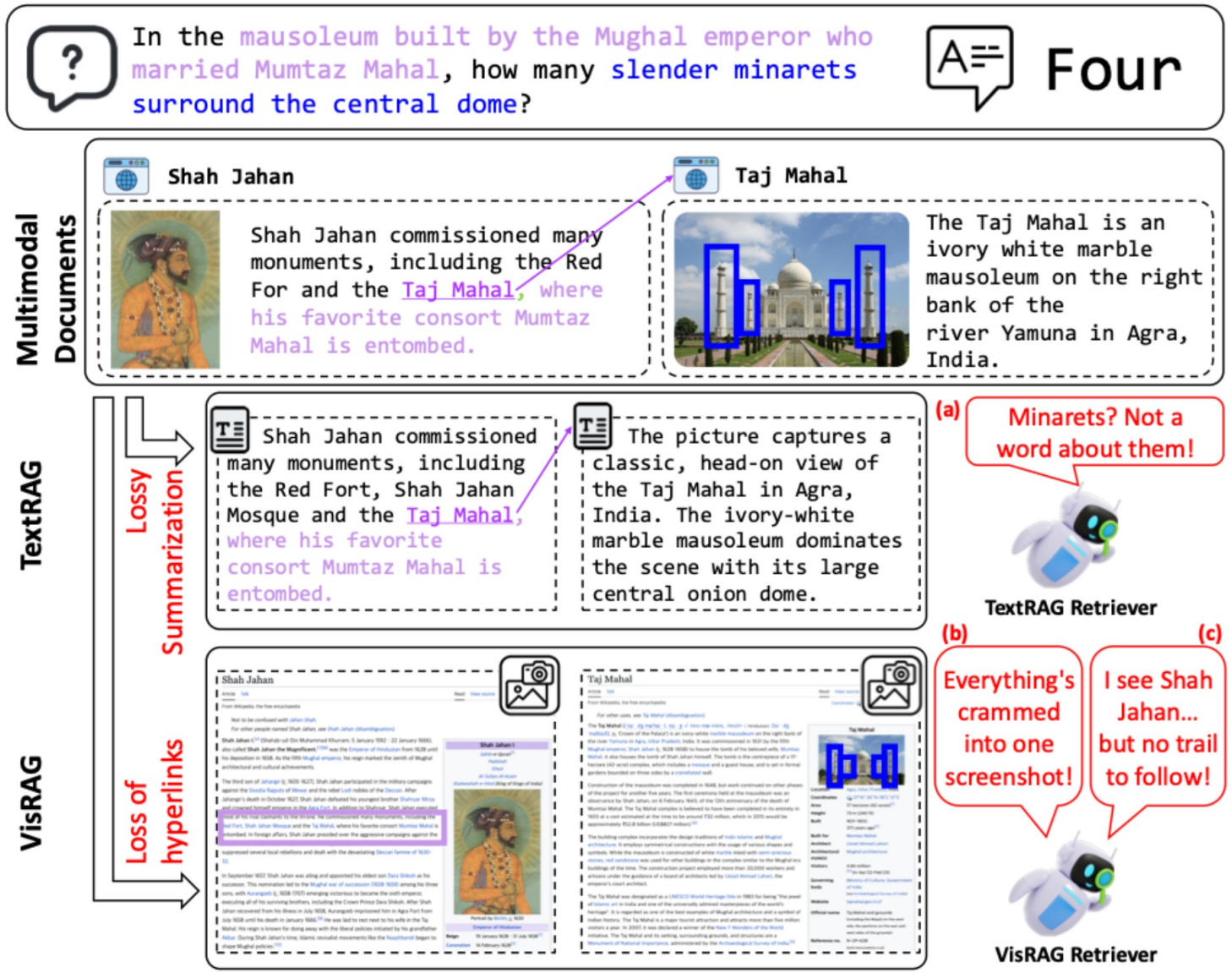
An example of an infographic

Open-domain Multimodal Document Retrieval

- Given a data source $\mathcal{D}$, a link mapping $\mathcal{L}$ and a query q, correctly retrieve a ranked list of components $= [C_1, \dots, C_{n_{ret}}]$. The goal for $\mathcal{R}$ is to contain the ground truth set of query-relevant components $= \{C_{gt_1}, \dots, C_{gt_r}\}$
- Data source $\mathcal{D} = \{D_1, \dots, D_{k_{doc}}\}$
  - Multimodal document $\mathrm{D} = [C_1, \dots, C_{k_{comp}}]$
    - Component $\mathrm{C} \in \{P, T, I\}$
    - Passage P : a natural language string
    - Table T : A matrix of natural language strings
    - Image I : A tensor of floating point numbers
- Link mapping $\mathcal{L} = \mathcal{C} \rightarrow \mathcal{D}$
  - $\mathcal{C}$: A set of all components within $\mathcal{D}$



Traditional Approaches: TextRAG and VisRAG



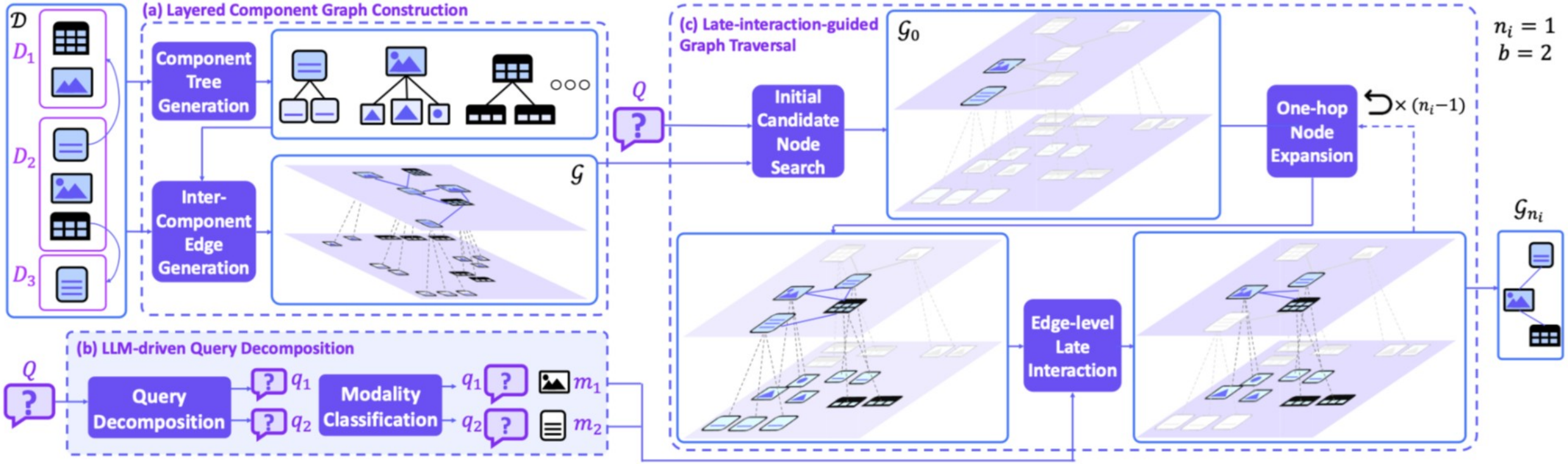
Key Aspects of Multimodal Document Retrieval

- **Preserve structure**: Model and exploit both inter-document links and intra-document relationships among components to maintain context and reasoning paths.
- **Modality awareness**: Identify which modality (text, table, figure/image) actually contains the evidence and route retrieval/scoring accordingly.
- **Granularity alignment**: Index and score at modality-appropriate levels—e.g., paragraph/sentence for text, table/row/cell for tables, and image/object/region for visuals.

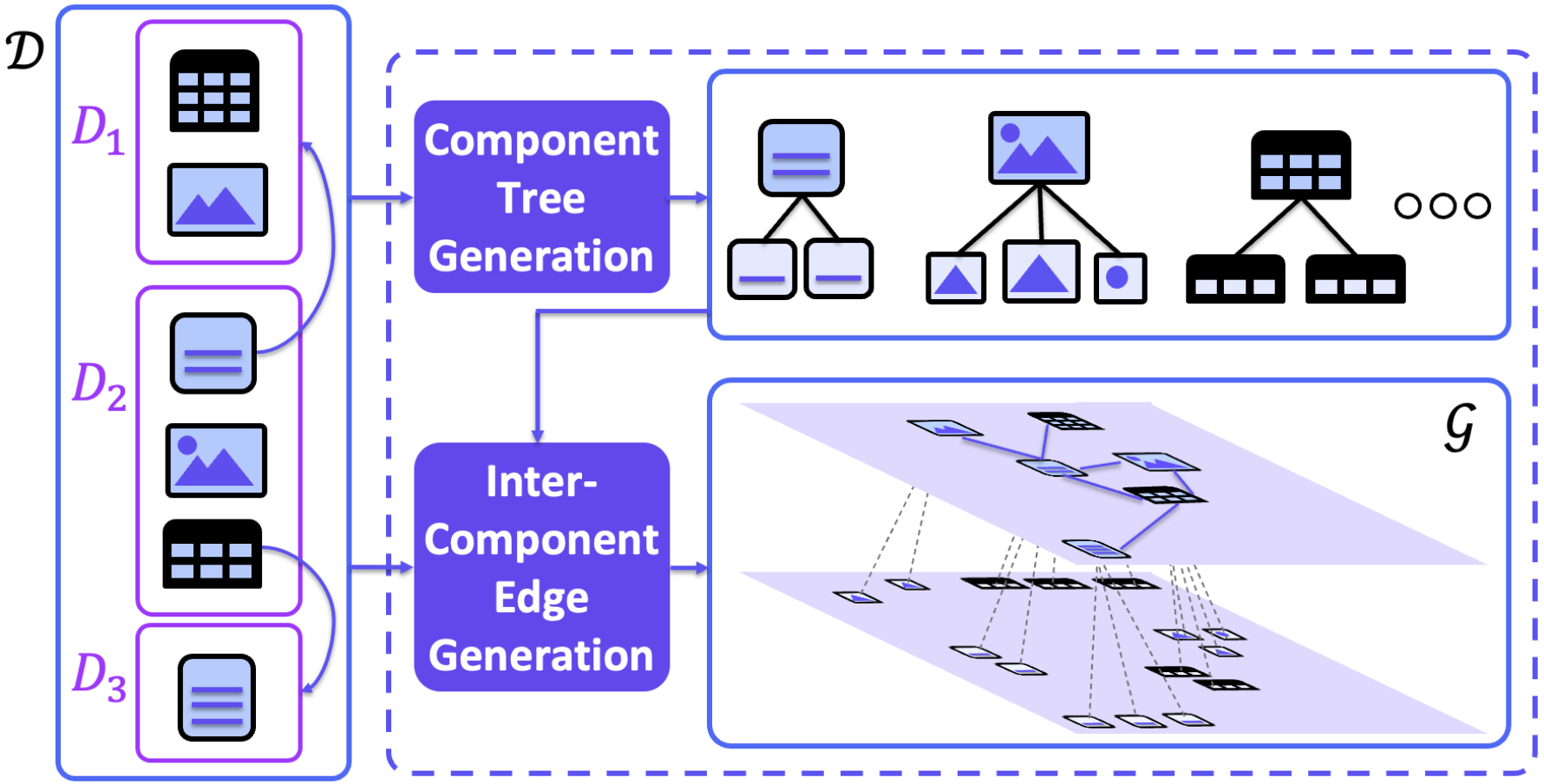
Key Ideas of Our Approach

- **Layered Component Graph**
  - Represents two granularities for each component.
    - Paragraph -> Sentences
    - Table -> Rows
    - Image -> Detected subimages
  - Expresses the link mapping and intra-document relationships as edges among components.
  - Uses multimodal embeddings to prevent any loss from modality conversion.
- **Retrieval Algorithm**
  - Fine-grained interaction with the graph and the query
    - Query decomposition
    - Late interaction

Overview of LILaC



Layered Component Graph Construction



Component Tree Generation

1. Break down components into subcomponents
   - Passage -> Split into sentences using SAT [1]
   - Table -> Split into rows
   - Image -> Detect subimages using MLLM
2. Connect each component with its subcomponents.

Inter-Component Edge Generation

Generate an undirected edge for

- Every pair of components within the same document.
- Every pair of component with its linked document's components.

[1] Frohmann, Markus, et al. "Segment Any Text: A Universal Approach for Robust, Efficient and Adaptable Sentence Segmentation." Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing. 2024.

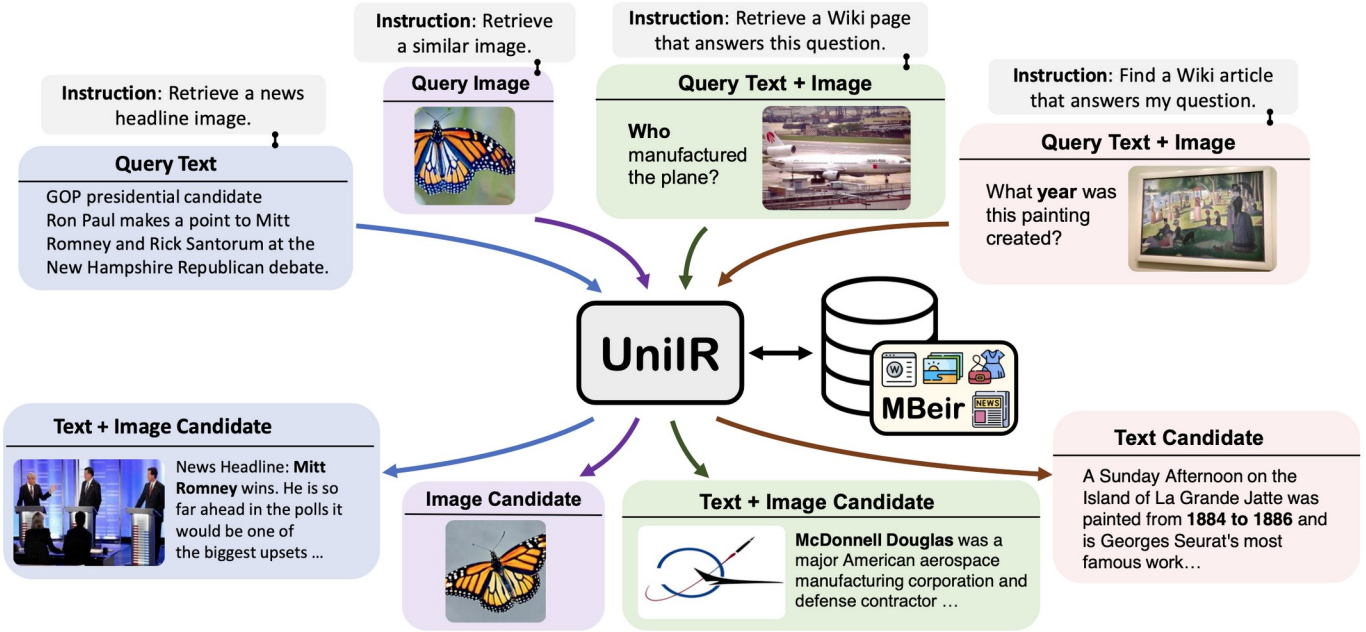
Background

Multimodal Embedders

Multimodal embedders embed a *natural language sequence*, an *image*, or a *combination of two* into a single vector.

Their retrieval setting includes an instruction for each retrieval.

- Instruction includes information about the target modality or target semantics of the retrieval.
- Such **instruction information is not given some cases** (e.g., our problem setting).



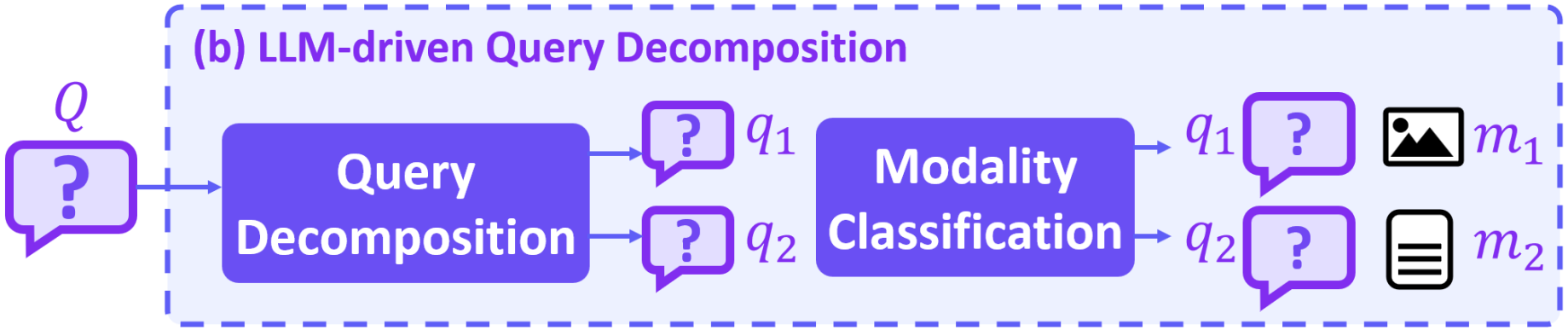
LLM-Driven Query Decomposition

A single query may include multiple evidences needed to locate multiple components. Such components may differ in their modalities.

Idea:

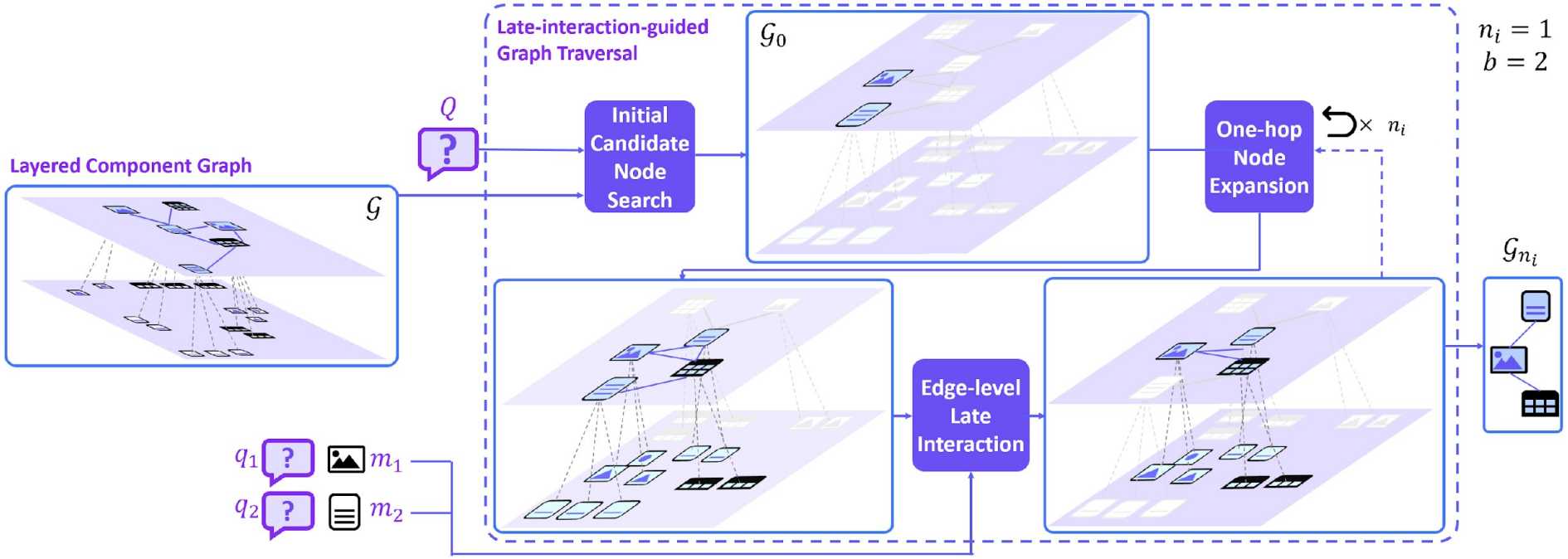
a. Decompose the given query so that each subquery contains fine-grained evidence for the target of retrieval.

b. Estimate (i.e., classify) the target modality for each subquery, and use it to generate an instruction for each subquery embedding.



Late-Interaction-Guided Graph Traversal

1. Initiate the graph $\mathcal{G}_0$ by performing kNN search over the coarse-grained components using the coarse-grained query Q.
2. Iteratively (for n_i times)
   2.1. Expand the graph by performing a one-hop node expansion.
   2.2. Leave top-b query-relevant edges among the expanded graph.



Edge-level Late Interaction

Goal: Similarity calculation of a query and an edge

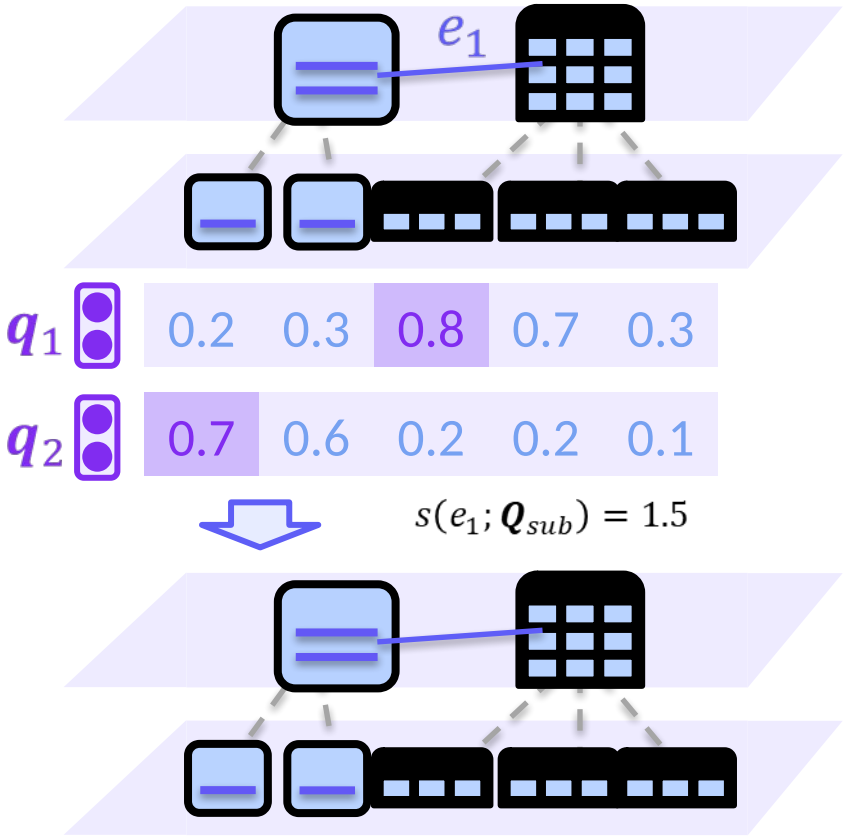
Input:

- Query embedding $\boldsymbol{Q}$
  - Subquery embeddings $\boldsymbol{Q_{sub}} = \{\boldsymbol{q_1}, \boldsymbol{q_2}, \dots, \boldsymbol{q_n}\}$
- Edge $e = (C_a, C_b)$
  - Subcomponent embeddings $\boldsymbol{e_c} = \{\boldsymbol{c_{a,1}}, \dots, \boldsymbol{c_{a,k}}\} \cup \{\boldsymbol{c_{b,1}}, \dots, \boldsymbol{c_{b,l}}\}$

Idea: Late-interaction-based score calculation

- A fine-grained subquery will show the highest cosine similarity with its corresponding target subcomponent.
- The edge similarity is the sum of all subcomponent scores.

$$s(e; \boldsymbol{Q_{sub}}) = \sum_{\boldsymbol{q} \in \boldsymbol{Q_{sub}}} \max_{\boldsymbol{c} \in \boldsymbol{e_c}} sim(\boldsymbol{c}, \boldsymbol{q})$$



Retriever Accuracy Comparison

- LILaC achieves state-of-the-art performance on all benchmarks.
- 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.

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	75.09	69.86	54.79	59.02	48.88	40.30
LILaC (w/ UniME)	Multimodal	77.83	71.42	84.35	77.93	82.28	75.27	58.52	61.44	49.63	42.97
LILaC (w/ MM-Embed)	Multimodal	**83.59**	**78.75**	**92.81**	**84.43**	**93.17**	**86.83**	**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.

End-to-End QA Accuracy Comparison

- LILaC achieves SOTA average end-to-end accuracy, with average EM and F1 scores of 52.00 and 58.79, respectively.
- 17.40% and 18.47% accuracy raise compared to the pre- 570 viously best-performing VisRAG setup, VisRAG 571 with Qwen2.5-VL

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	50.12	53.12	40.72	47.46	33.90	40.38
LILaC (w/ UniME)	Qwen2.5-VL 7B	62.43	69.40	53.05	62.89	53.39	56.86	43.42	49.72	33.39	40.12
LILaC (w/ MM-Embed)	Qwen2.5-VL 7B	**65.48**	**72.42**	**55.57**	**66.32**	**60.91**	**62.87**	**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.

Thank You

Experiment Settings

Benchmarks:

- **DocVQA Benchmarks**
  - InfoVQA
  - MP-DocVQA
  - SlideVQA
- **Multimodal QA Benchmarks**
  - MultimodalQA
  - MMCoQA

Competitors

- VisRAG (ICLR'25)
- M3DocRAG (ArXiv'24)

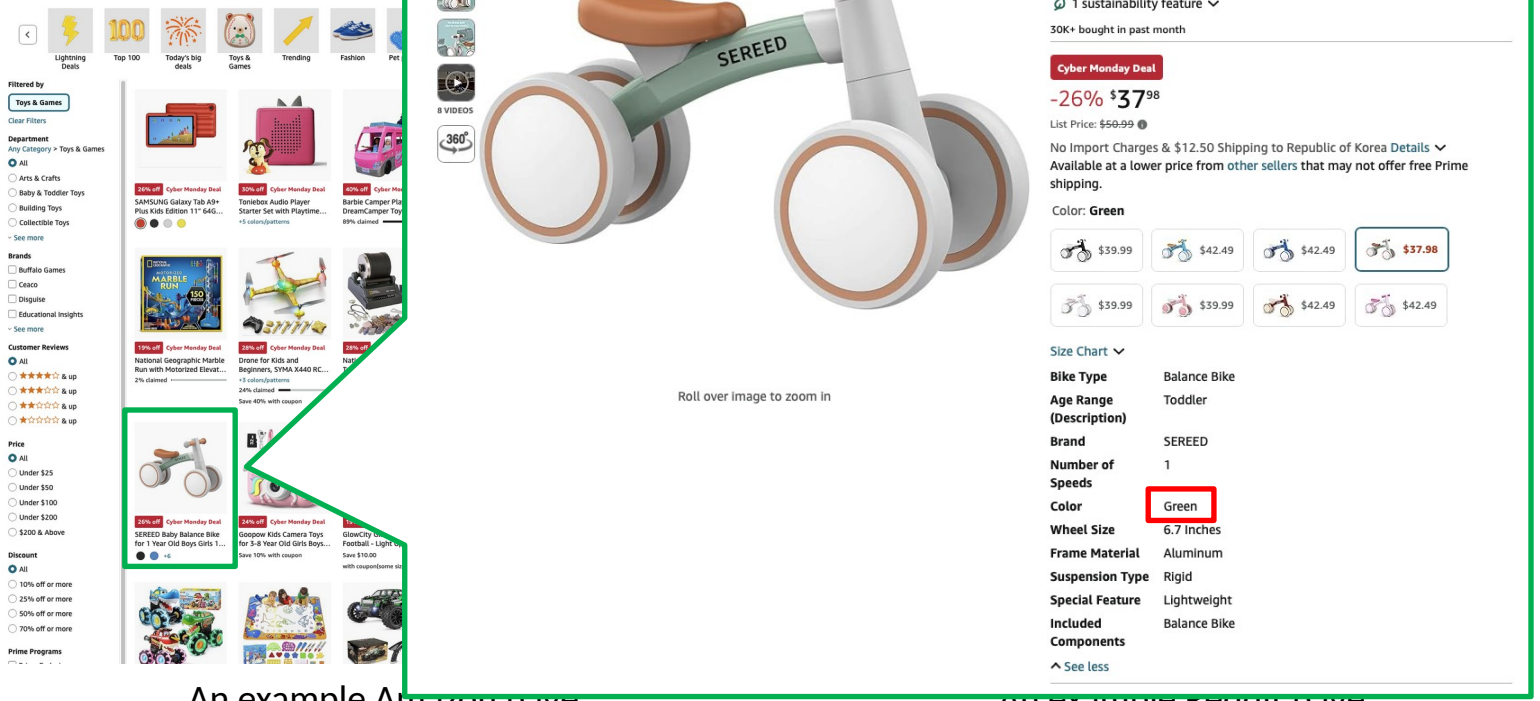
Measurements

- **Component retrieval accuracy**
  - Recall@3
  - MRR@10
- **End-to-end accuracy**
  - Exact match (EM)
  - F1 score

Introduction

Multimodal (Text & Image) Data Source

- Most of real-world data are comprised of text and images.
- Images can play an irreplaceable role in answering some questions.
  - e.g., “Recommend me a white baby bike”
- However, most of prior works do not consider ir

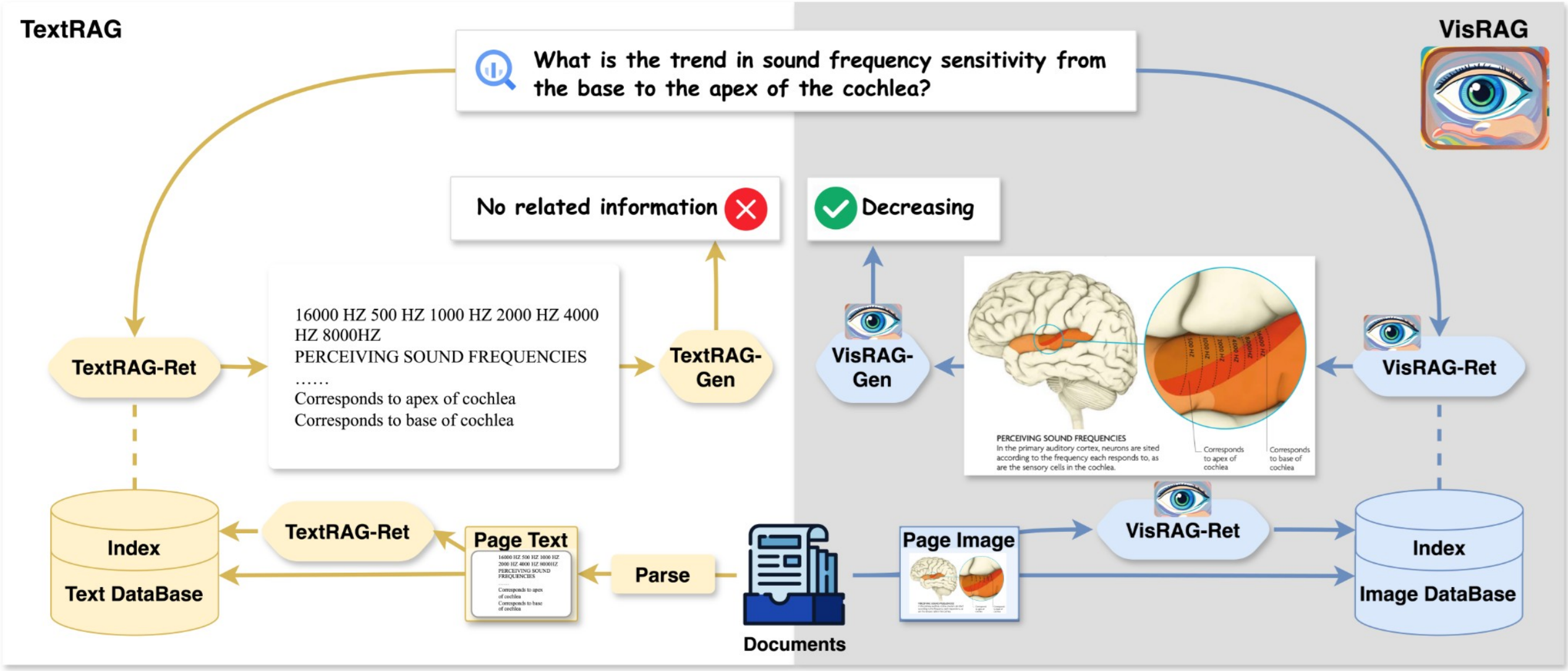


A example Wikipedia page

An example Amazon page

An example Reddit page

Traditional Approaches: TextRAG and VisRAG



Ablation Study

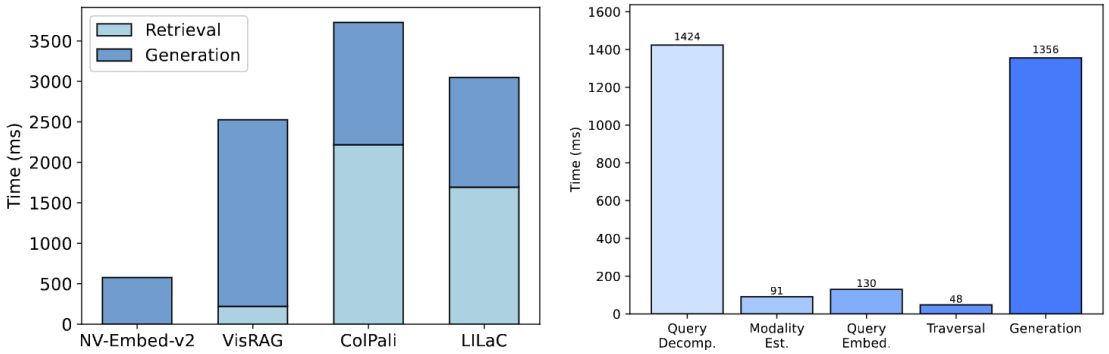
- Employing the layered graph structure leads to notable average improvements - 7.33% in R@3 and 10.13% in MRR@10.
- Further employing query decomposition & late interaction shows incremental gains of 3.19% in R@3 and 4.7% in MRR@10.

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	LILaC ((w/o Query Decomposition) & Layered Graph)	48.90	43.97	75.91	68.13	65.60	58.55	42.99	46.92	41.22	34.51
	LILaC (w/o Query Decomposition)	60.81	55.02	74.14	67.58	67.70	60.01	45.15	51.12	44.18	36.62
	LILaC	**61.25**	**55.35**	**76.80**	**68.99**	**68.91**	**61.18**	**54.78**	**59.32**	**48.54**	**40.22**
UniME	LILaC ((w/o Query Decomposition) & Layered Graph)	52.12	45.31	81.47	71.22	83.57	77.07	47.68	49.06	45.78	38.41
	LILaC (w/o Query Decomposition)	77.83	71.27	83.45	75.70	85.11	78.01	52.18	54.01	47.11	39.85
	LILaC	**77.83**	**71.39**	**84.35**	**77.93**	**85.53**	**78.81**	**58.43**	**61.32**	**49.45**	**42.91**
MM-Embed	LILaC ((w/o Query Decomposition) & Layered Graph)	75.80	69.09	92.80	82.19	90.39	83.71	61.10	67.35	47.94	43.75
	LILaC (w/o Query Decomposition)	82.23	77.75	92.27	83.20	92.17	85.53	63.19	69.91	50.18	45.59
	LILaC	**83.59**	**78.75**	**92.81**	**84.43**	**93.17**	**86.83**	**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.

Algorithm Runtime Analysis

- LILaC is approximately 20.76% slower than VisRAG, yet 18.24% faster than ColPali.
- 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 de- composition phase, averaging 1,423 ms.



(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.

Next Research

Possible research directions

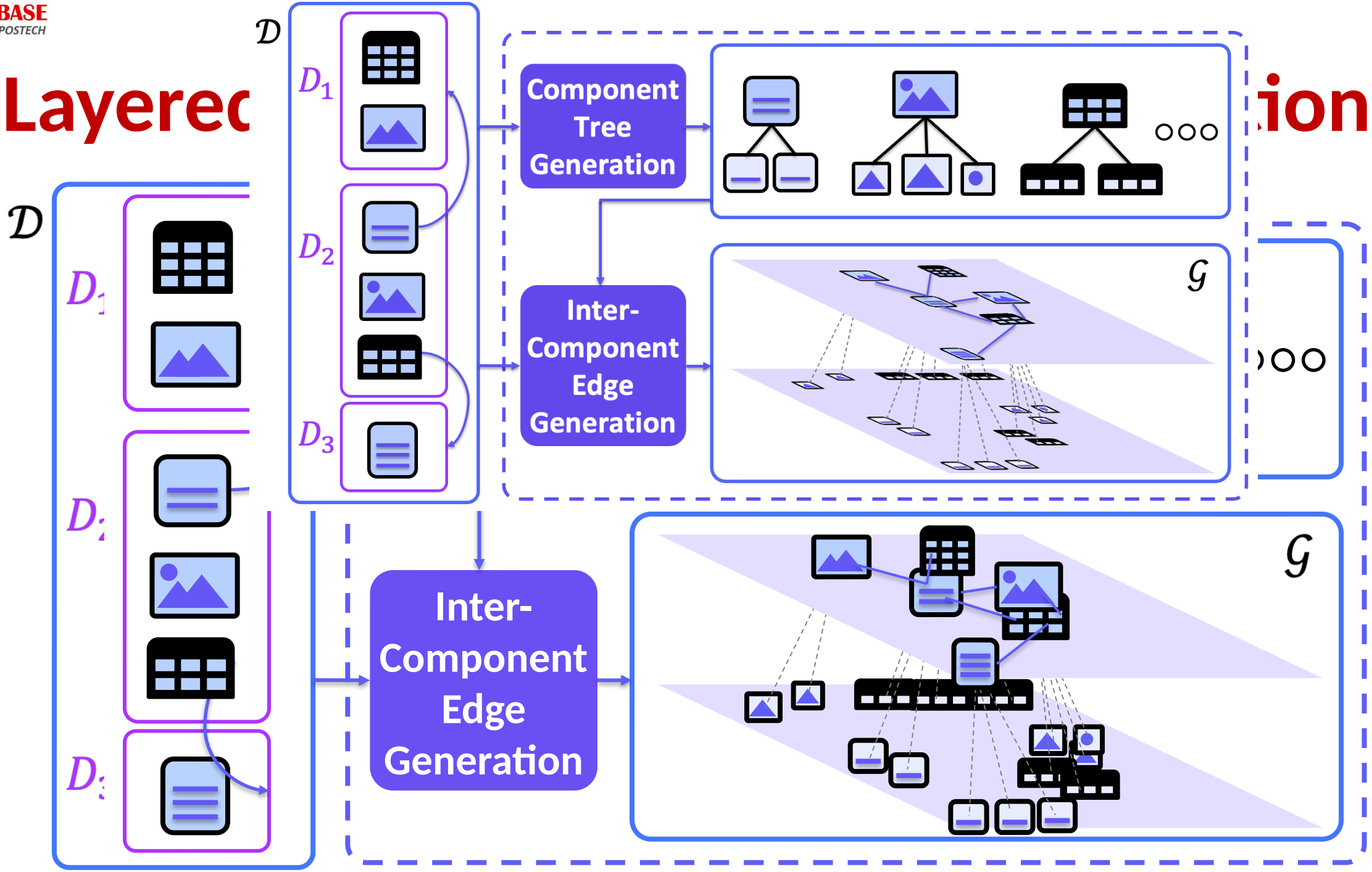
- Problem generalization
  - Expand the problem beyond Wikipedia setting
- Multimodal retrieval improvement
  - Advanced graph construction
  - LLM-guided retrieval
- Multimodal generation improvement

Current theoretical problems of LILaC

- Cannot retrieve components for which dependency resolution is needed.
- Ineffective to retrieve long-hop evidences.

-> Currently performing a fine-grained error analysis.

Layerec ion


$\mathcal{D}$
D_1
D_2
D_3
Component Tree Generation
Inter-Component Edge Generation
$\mathcal{G}$
Inter-Component Edge Generation
$\mathcal{G}$