

Open-Domain Multimodal Multihop Retrieval

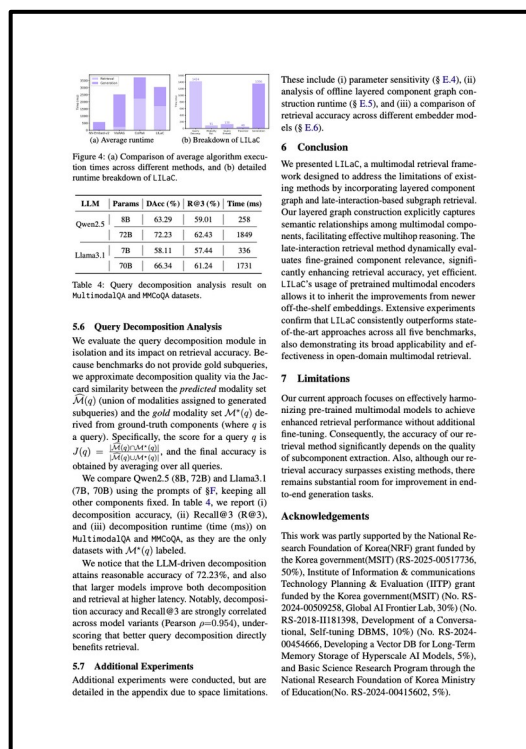
Data Structure & Agentic Retrieval Algorithm

2026.5.28

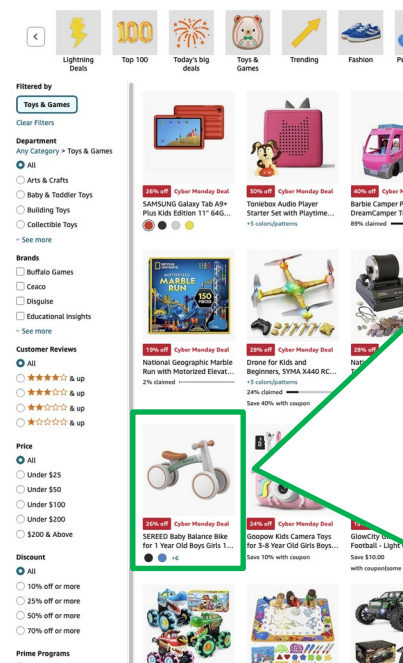
Presenter: Joohyung Yun

Multimodal Documents Are Everywhere!

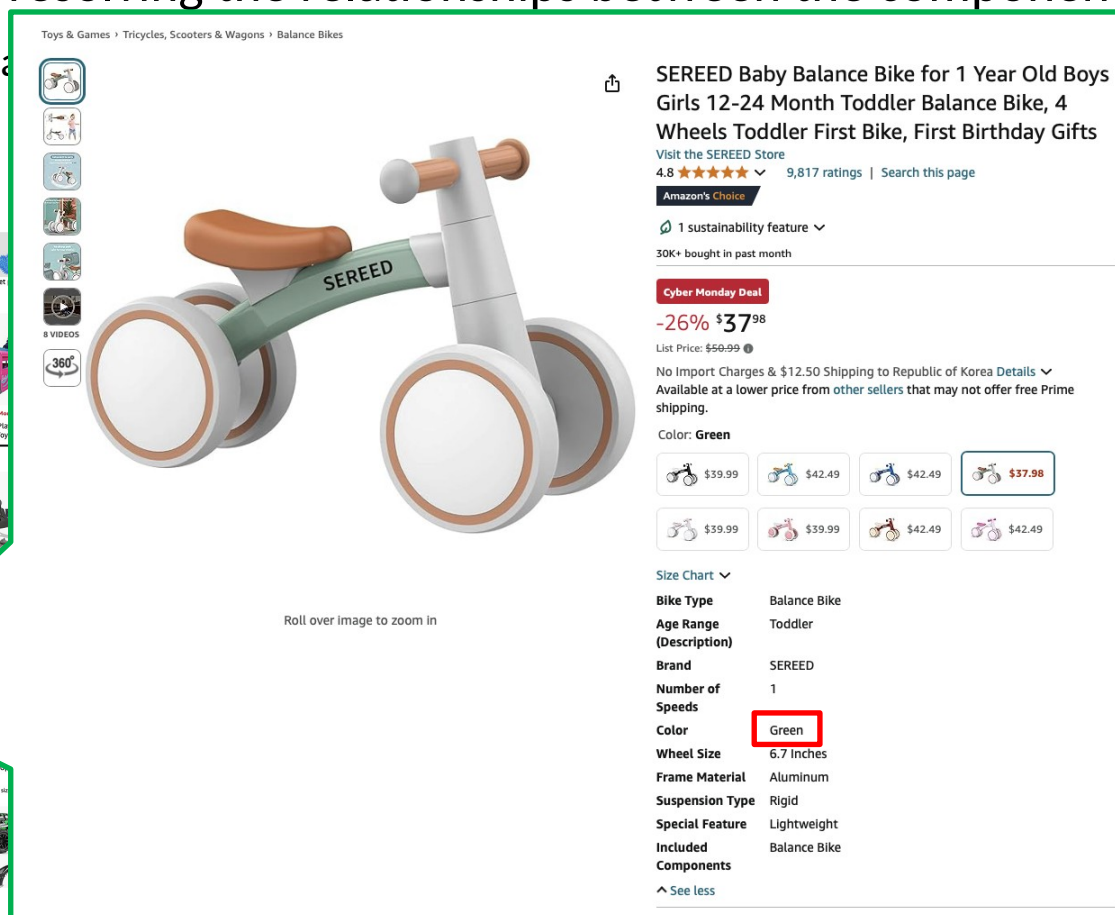
- Real-world documents are often multimodal.
 - e.g., webpages, PDFs, presentation slides, and way more!
- Accurate question-answering involves correctly resolving the relationships between the components.
 - Component*: a multimodal element (i.e., text/table/image)



An example PDF file



An example Amazon page



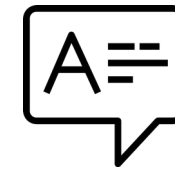
An example Reddit page

Multimodal Document Question Answering

Correctly answering the given question using information of multimodal components



In the mausoleum built by the Mughal emperor who married Mumtaz Mahal, how many slender minarets surround the central dome?

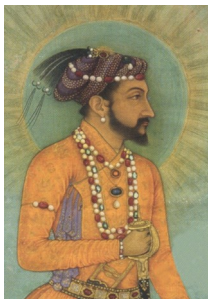


Four

Multimodal
Documents



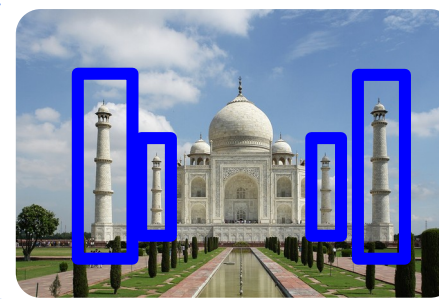
Shah Jahan



Shah Jahan commissioned many monuments, including the Red Fort and the Taj Mahal, where his favorite consort Mumtaz Mahal is entombed.



Taj Mahal



The Taj Mahal is an ivory white marble mausoleum on the right bank of the river Yamuna in Agra, India.

Brief Guide to Today's Seminar

- LLaC: Late Interacting in Layered Component Graph for Open-domain Multimodal Multihop Retrieval (EMNLP'25)
 - **Main Question:** What is an effective way to capture the characteristics of multimodal documents? What is an appropriate data structure?
- Failure is Feedback: History-Aware Backtracking for Agentic Traversal in Multimodal Graphs (ICML'26)
 - **Main Question:** What is an effective way to perform traversal on such kind of data structure?



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

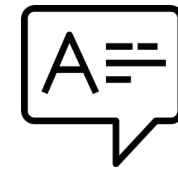
Joohyung Yun, Doyup Lee, and Wook-Shin Han*

Multimodal Document Question Answering (Revisited)

Correctly answering the given question using information of multimodal components



In the mausoleum built by the Mughal emperor who married Mumtaz Mahal, how many slender minarets surround the central dome?

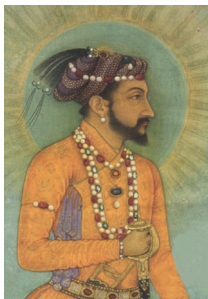


Four

Multimodal
Documents



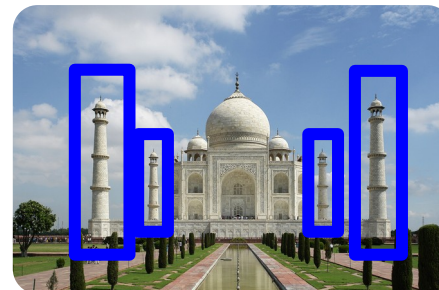
Shah Jahan



Shah Jahan commissioned many monuments, including the Red Fort and the Taj Mahal, where his favorite consort Mumtaz Mahal is entombed.



Taj Mahal



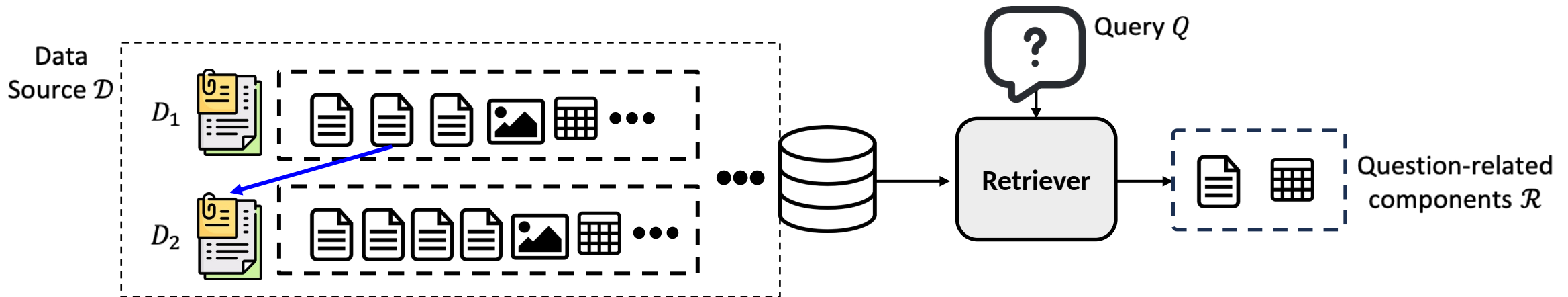
The Taj Mahal is an ivory white marble mausoleum on the right bank of the river Yamuna in Agra, India.

Our Task

Open-Domain Multimodal Document Retrieval

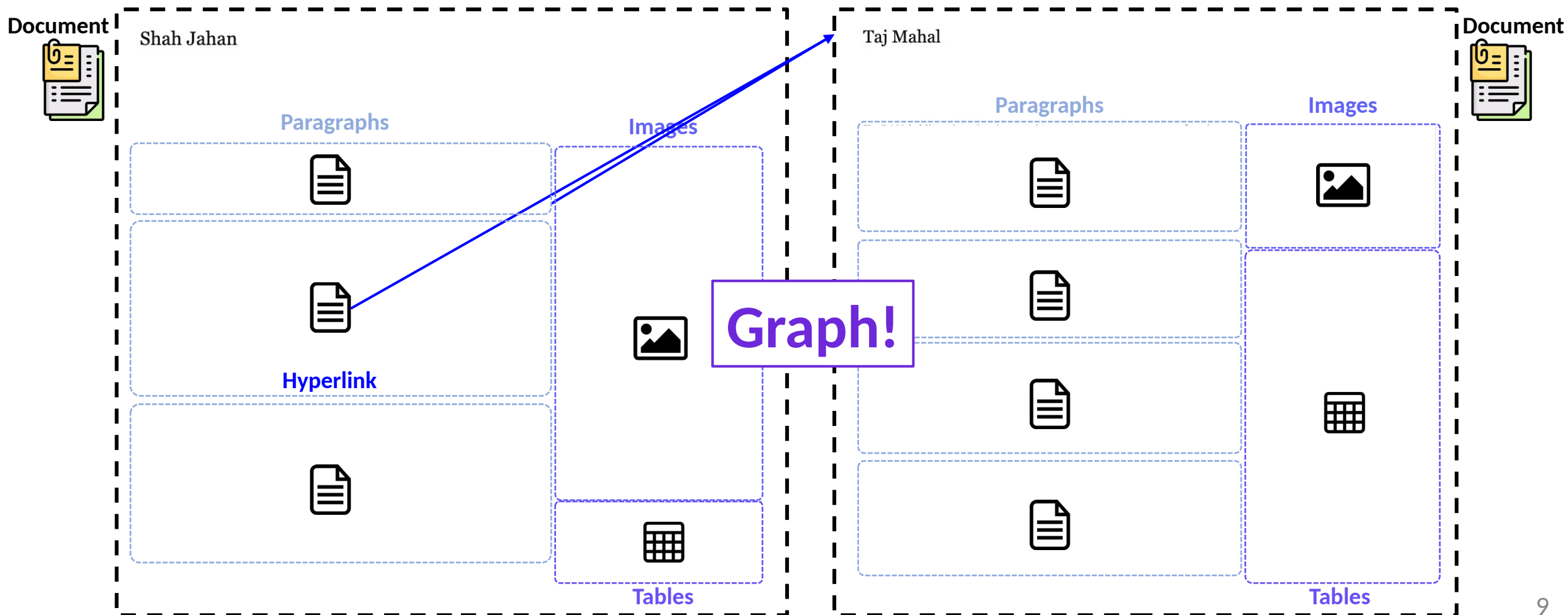
Given a data source $\mathcal{D}$, a link mapping $\mathcal{L}$ and a query Q , correctly retrieve a ranked list of components $\mathcal{R} = [C_1, \dots, C_{n_{ret}}]$. $\mathcal{R}$ should contain the ground truth set of components $\{C_{gt_1}, \dots, C_{gt_r}\}$.

- Data source $\mathcal{D} = \{D_1, \dots, D_{k_{doc}}\}$
- Document $D = \{C_1, \dots, C_{k_{comp}}\}$
- Component $C \in \{P, T, I\}$
 - P : paragraph, T : table, I : image
- Link mapping $\mathcal{L} = \mathcal{C} \rightarrow \mathcal{D}$
 - $\mathcal{C}$: A set of all components within $\mathcal{D}$



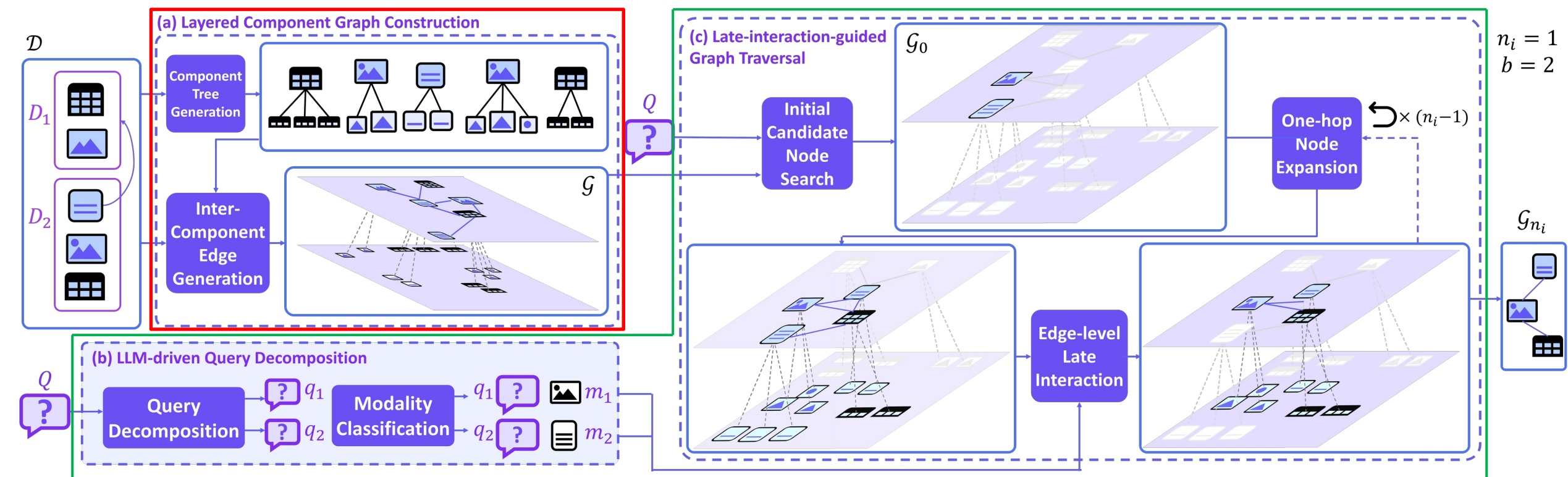
How Can We Model Multimodal Documents?

- Component: One of paragraph, table or an image
- Document: A set of components



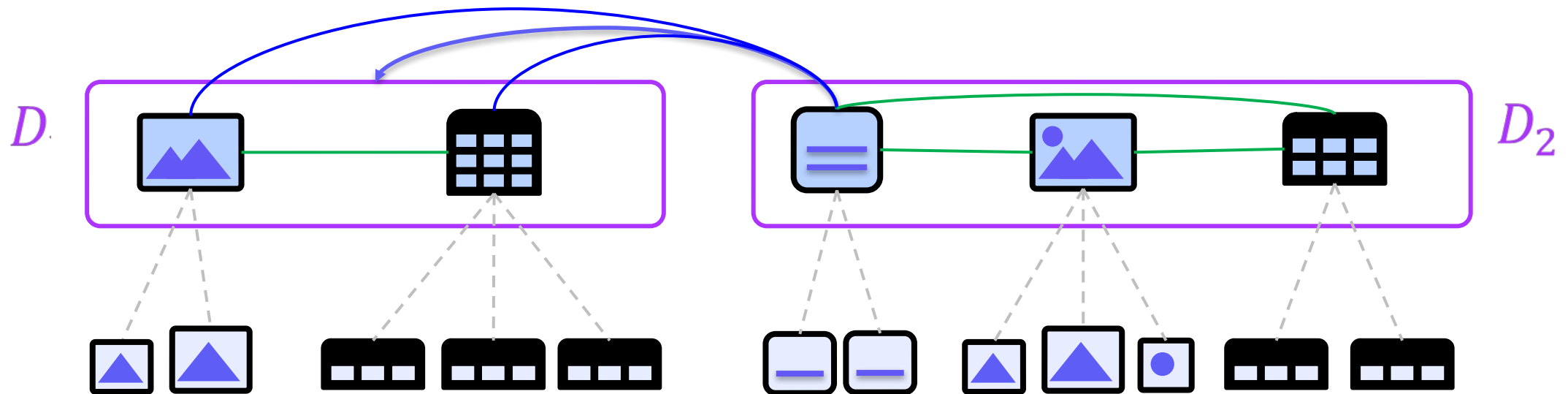
Overview of LLaC

- **Offline:** generate a graph data structure
- **Online:** retrieve a query-relevant subgraph from the graph



What Should We Consider for the Graph?

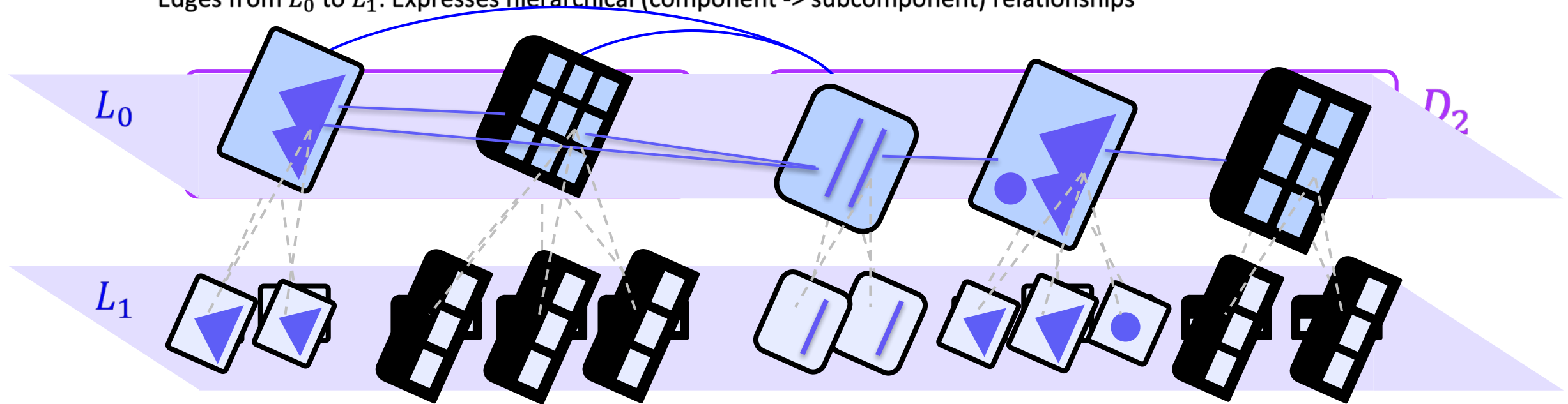
- Modality preservation
- Multimodal multihop reasoning: Exploit both **inter-document links** and **intra-document relationships**
- Consideration of fine-grained granularity [1]: Paragraph -> sentences, Table -> rows, Image -> detected objects



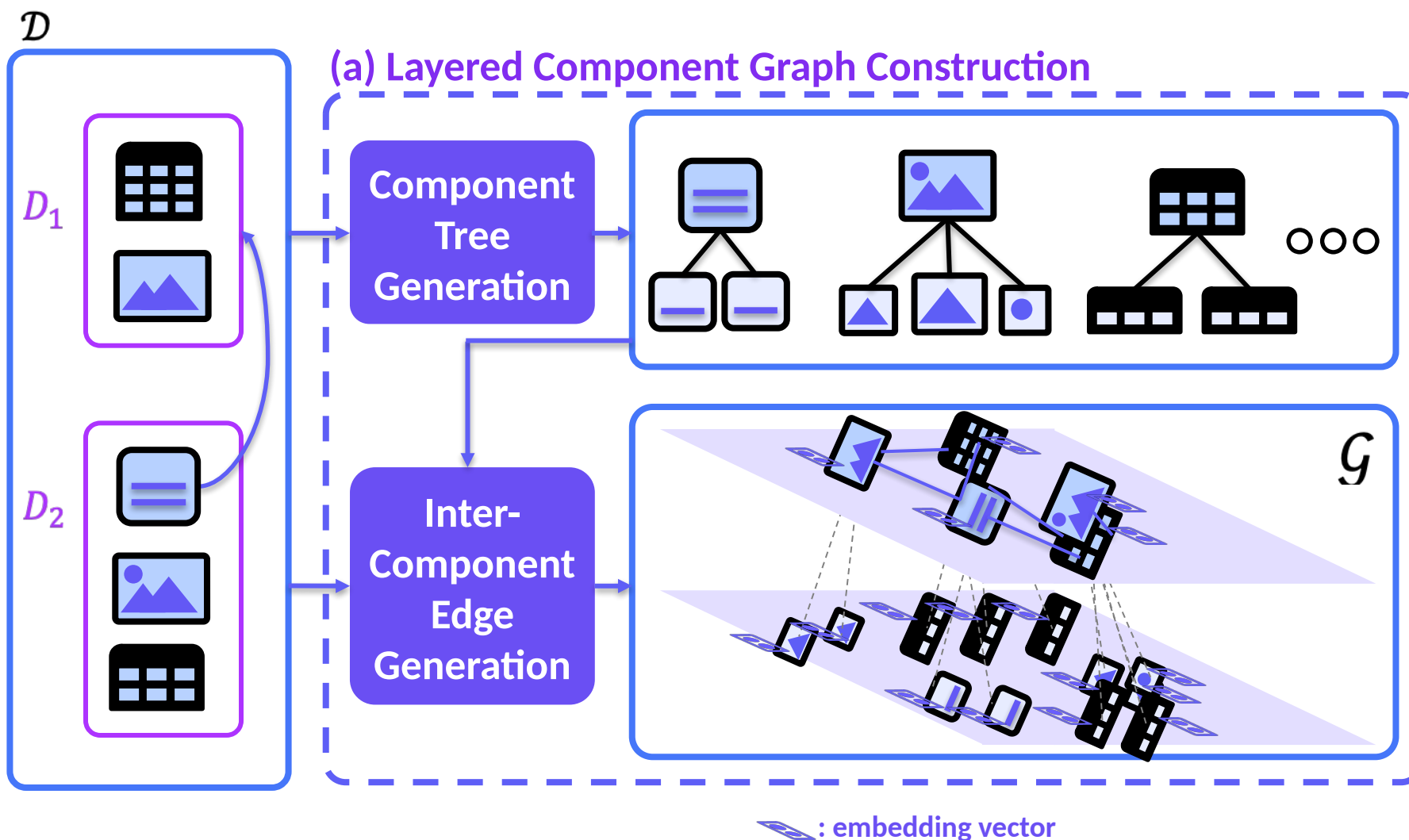
Key Data Structure

Layered Component Graph $\mathcal{G}$

- **Node:** component or subcomponent (e.g., sentence, row, image detection)
 - **Two layers:** each layer contains nodes for different granularity (coarse-grained L_0 vs fine-grained L_1)
- **Edges:**
 - Edges within L_0 : Expresses the hyperlink relationship and intra-document relationships
 - Edges from L_0 to L_1 : Expresses hierarchical (component $\rightarrow$ subcomponent) relationships



Layered Component Graph Construction



Component Tree Generation

1. Break down components into subcomponents
 - Passage -> Split into sentences
 - Table -> Split into rows
 - Image -> Detect subimages
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.

Embedding Generation

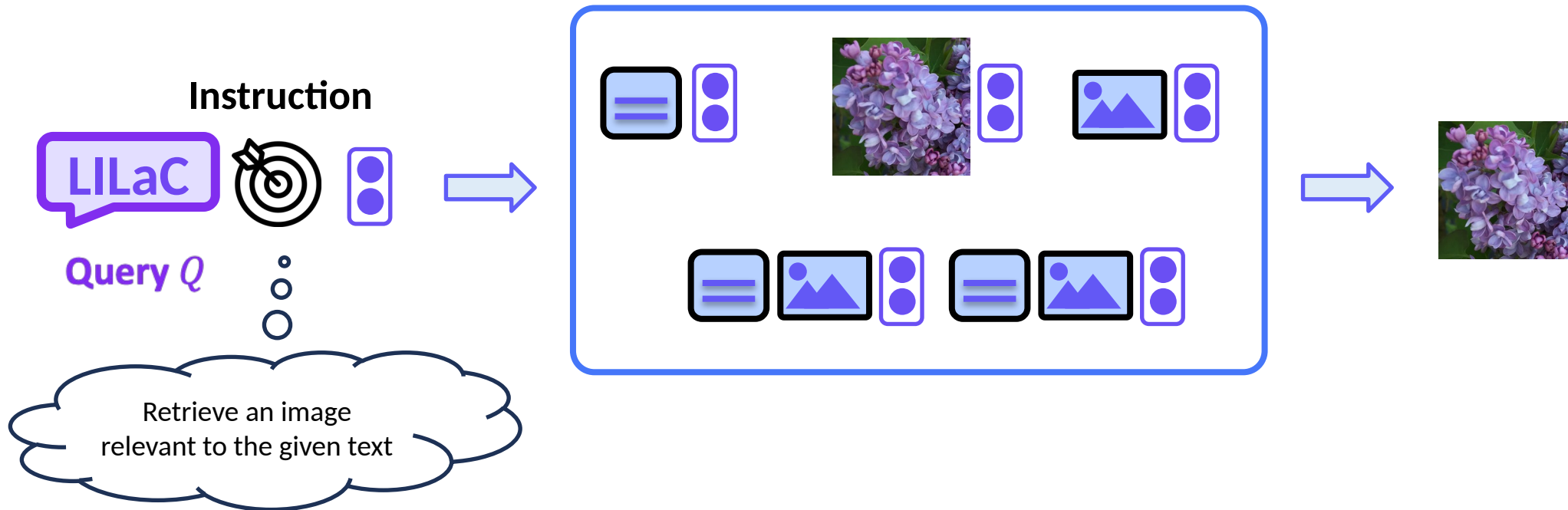
Background

Multimodal Embedders

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

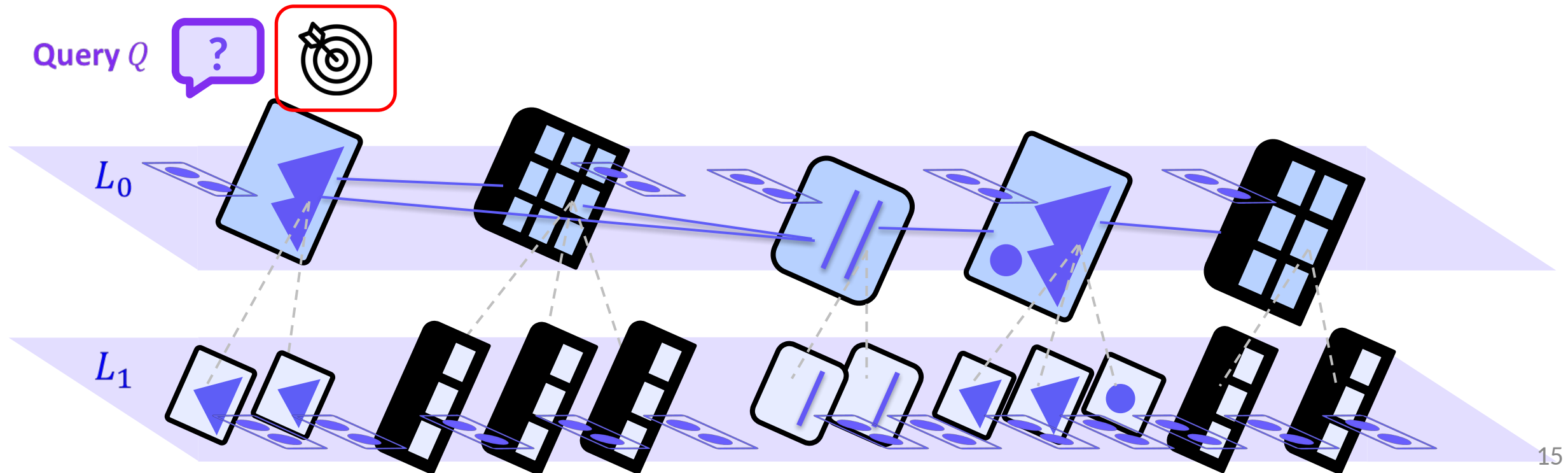
They require an instruction for each retrieval.

- Instruction includes information about the *target modality* of the retrieval.



What Should We Consider for the Retrieval Algorithm?

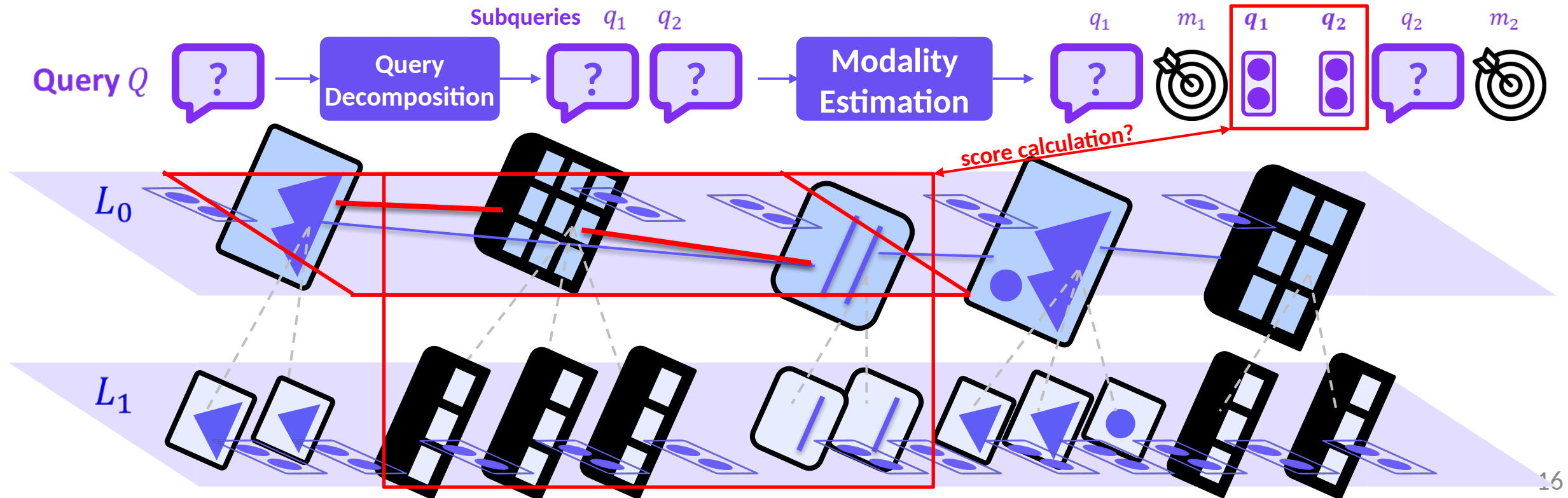
- How should we find the instructions for multimodal embedding?
- **Challenge 1:** The instructions are not given in our setting
- How should we find subgraphs that need multimodal multihop reasoning?
- **Challenge 2:** The target subgraph may consist of multiple multimodal nodes (multihop reasoning)



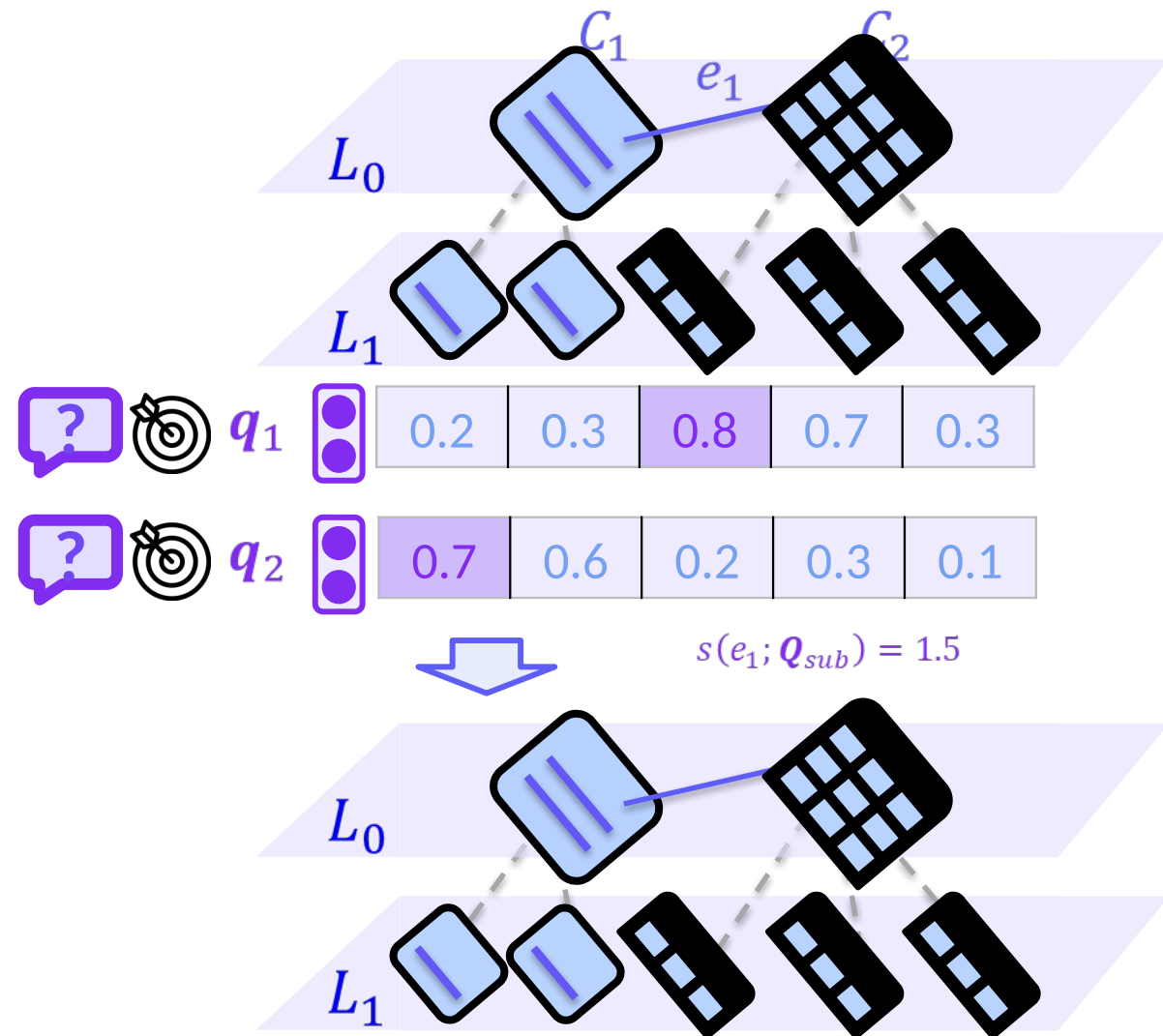
Key Idea

LILaC Retrieval Algorithm

- **Query decomposition & modality estimation:** extract subqueries, then perform modality estimation for each subquery.
- **Edge-based subgraph retrieval:** retrieves query-relevant edges, then merges them into a subgraph



Edge-level Late Interaction



Goal: Similarity calculation of a query and an edge

Input:

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

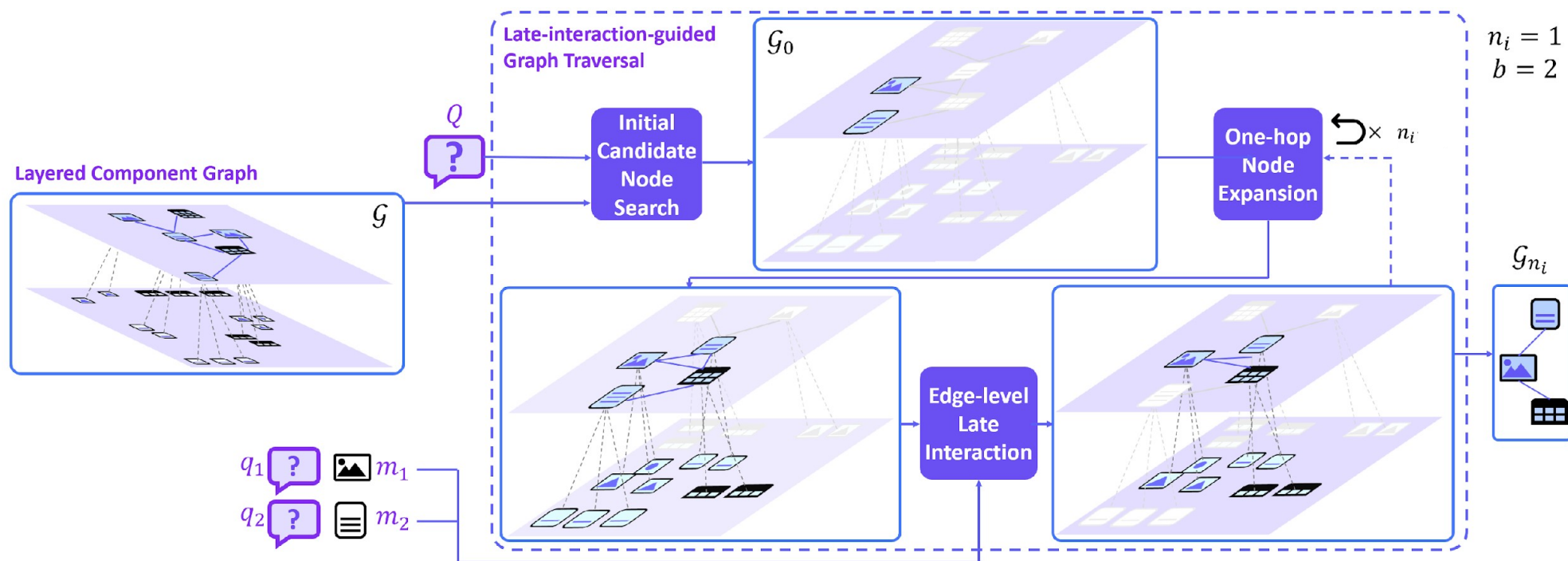
Motivation: A subquery is likely to show the highest similarity with its corresponding golden subcomponent.

Late-interaction-based score calculation:

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

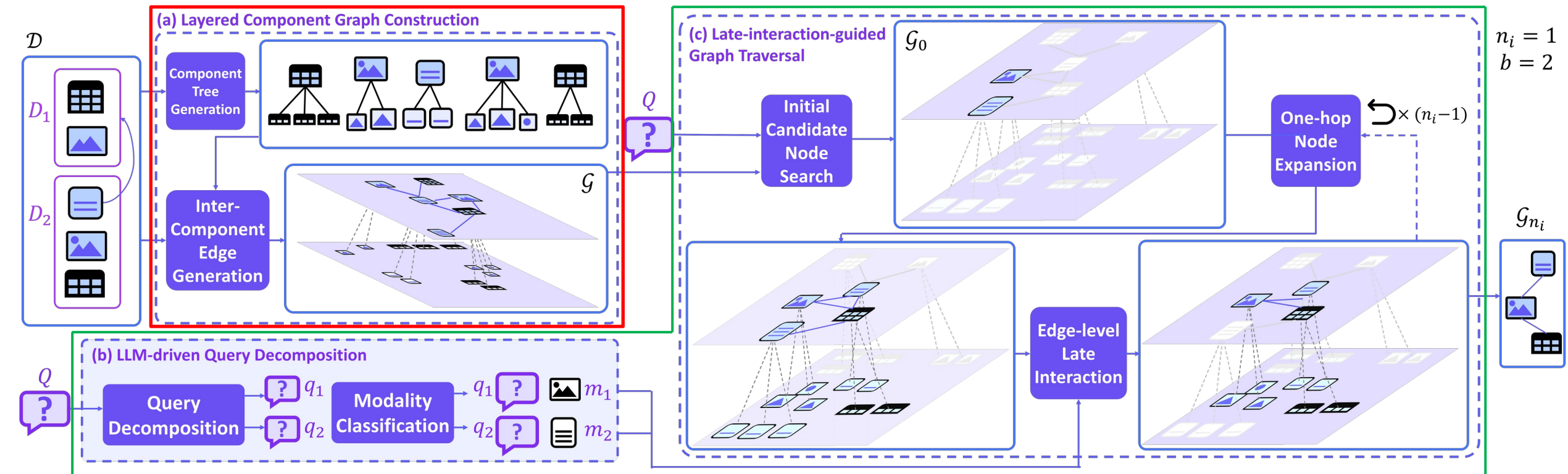
Late-Interaction-Guided Graph Traversal

1. Initiate the subgraph by performing kNN search over components
2. Iteratively
 - a. Expand the subgraph by performing a one-hop node expansion from existing nodes.
 - b. Leave top- b query-relevant edges among the expanded graph.



Overview of LLaC (Revisited)

- **Offline:** generate a graph data structure
- **Online:** retrieve a query-relevant subgraph from the graph



Retriever Accuracy Comparison

- LILaC achieves SOTA performance on all 5 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		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)		77.83	71.42	84.35	77.93	82.28	75.27	58.52	61.44	49.63	42.97
LILaC (w/ MM-Embed)		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 52.57 and 59.36, respectively.
- 17.40% and 18.47% accuracy raise compared to the previously best-performing VisRAG 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.

LILaC: What We Built, What's Left



CONTRIBUTION

What LILaC Built

A novel data structure

modality-preserving · fine-grained

The Layered Component Graph — modality-typed components (text / table / image), linked by hierarchical & navigational edges.

Effective and efficient retrieval

SOTA on multimodal multi-hop

Edge-level late interaction delivers +17.40% EM and +18.47% F1 over the prior best (VisRAG).



WHAT REMAINS

A Static Traversal

Decomposition is fixed once

no re-planning

Sub-queries are planned once at the start and never revised — even when a hop clearly fails.

Edges scored one way

embedding-only scoring

Relevance is judged by a single embedding-based rule — no hop-specific reasoning when similarity misleads.

→ The traversal is still rigid & one-shot.

THE BRIDGE TO OUR SECOND WORK

FIF keeps the **same graph**, but rethinks **how to traverse it** — making retrieval adaptive and failure-aware.

NEXT

Failure is Feedback



Failure is Feedback: History-Aware Backtracking for Agentic Traversal in Multimodal Graphs

Joohyung Yun, Doyup Lee, and Wook-Shin Han*

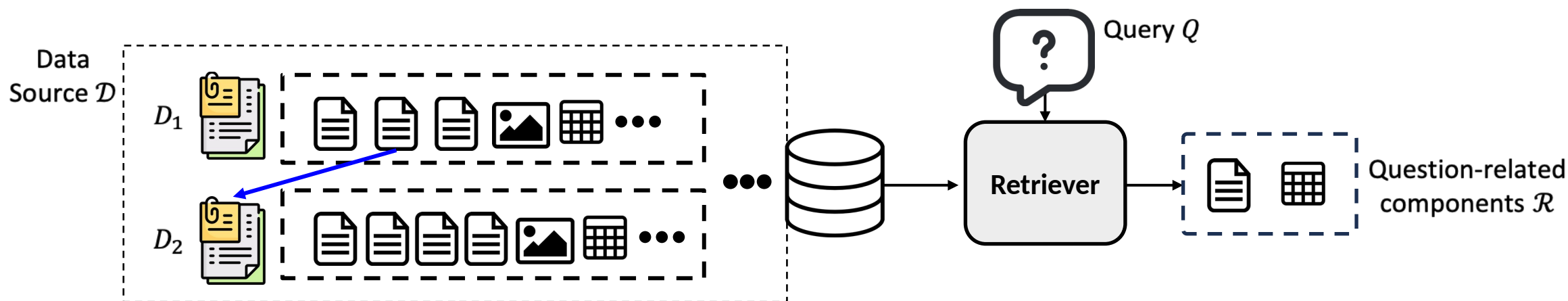


Our Task (Revisited)

Open-Domain Multimodal Document Retrieval

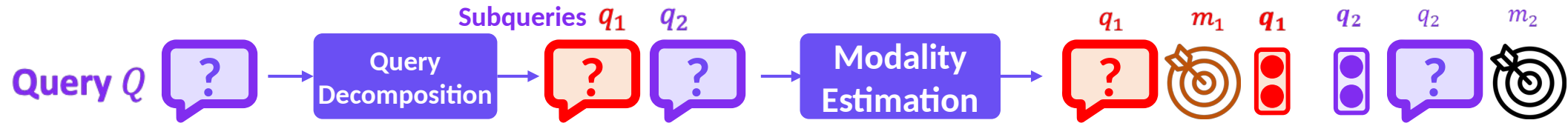
Given a data source $\mathcal{D}$, a link mapping $\mathcal{L}$ and a query Q , correctly retrieve a ranked list of components $\mathcal{R} = [C_1, \dots, C_{n_{ret}}]$. $\mathcal{R}$ should contain the ground truth set of components $\{C_{gt_1}, \dots, C_{gt_r}\}$.

- Data source $\mathcal{D} = \{D_1, \dots, D_{k_{doc}}\}$
- Document $D = \{C_1, \dots, C_{k_{comp}}\}$
- Component $C \in \{P, T, I\}$
 - P : paragraph, T : table, I : image
- Link mapping $\mathcal{L} = \mathcal{C} \rightarrow \mathcal{D}$
 - $\mathcal{C}$: A set of all components within $\mathcal{D}$



Limitations of LLaC – 1

Static Decomposition of Queries



? Was the **Taifun rocket system** used at the Battle of Amiens?

A≡ No

Decomposed Queries



What **battles** were the **Taifun rocket** system used in?



Is there Battle of Amiens in **such battles**?

Documents



Taifun Rocket

Taifun (German for "typhoon") was a German World War II anti-aircraft unguided rocket system.



Shepard Fairey



The Battle of Amiens was the opening phase of the Allied offensive, and happened during the World War I.



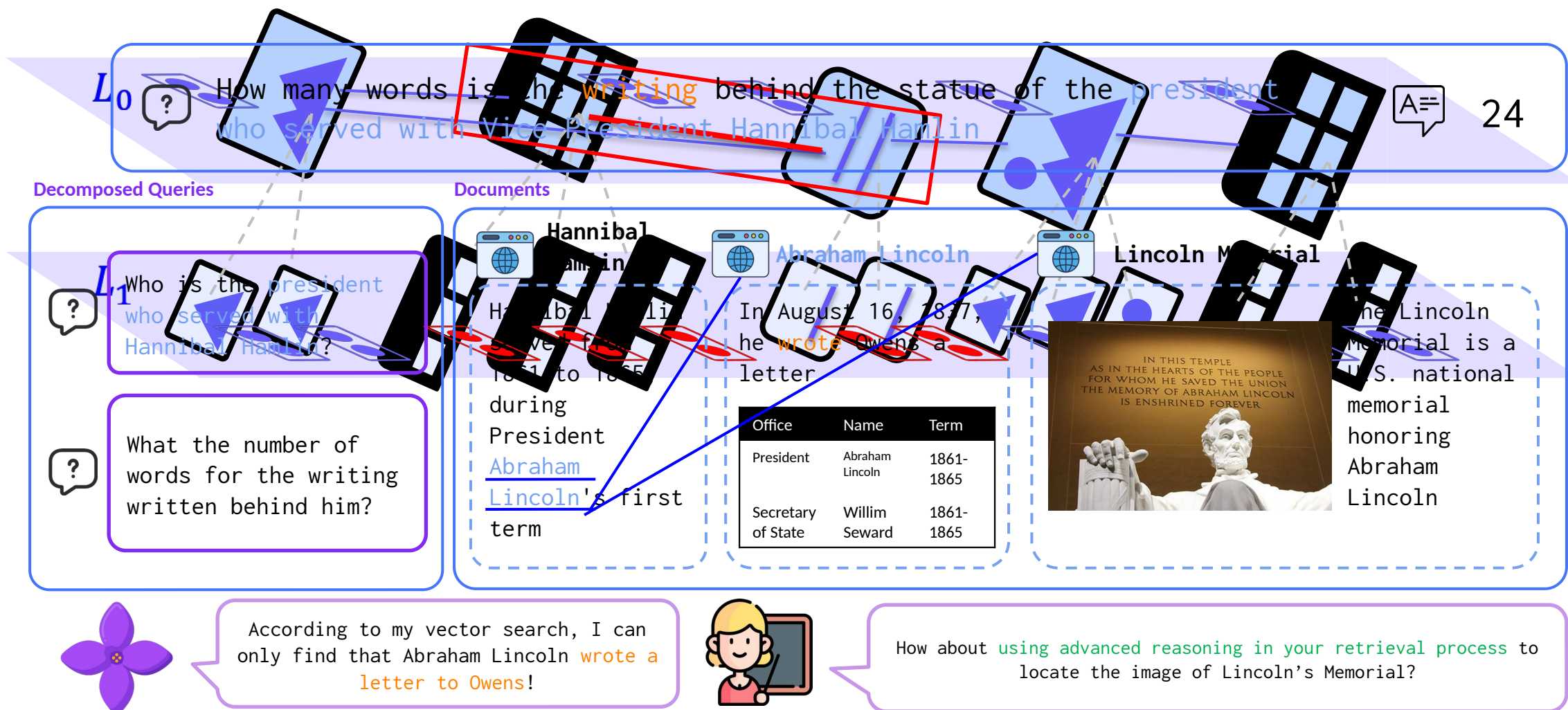
I can't find what **such battles** are! What are they?



How about **adjusting your plan using retrieved components** to retrieve more information about the Battle of Amiens first?

Limitations of LLaC - 2

Embedding-Based-Only Edge Interpretation



Key Idea: Adaptive Retrieval

STATIC · LILaC

One scoring rule, always

embedding
similarity only

*same cheap
rule for all*

ADAPTIVE

escalate on fail

Vector only

Vector on
finer granularity

+ LLM rerank

STATIC · LILaC

decide once

Query Q

q₁ (fixed)

q₂ (fixed)

*never changes,
even if a hop fails*

ADAPTIVE · FIFO

revise as you go

Query Q

q₁ ✓

q₂ ✗ fail

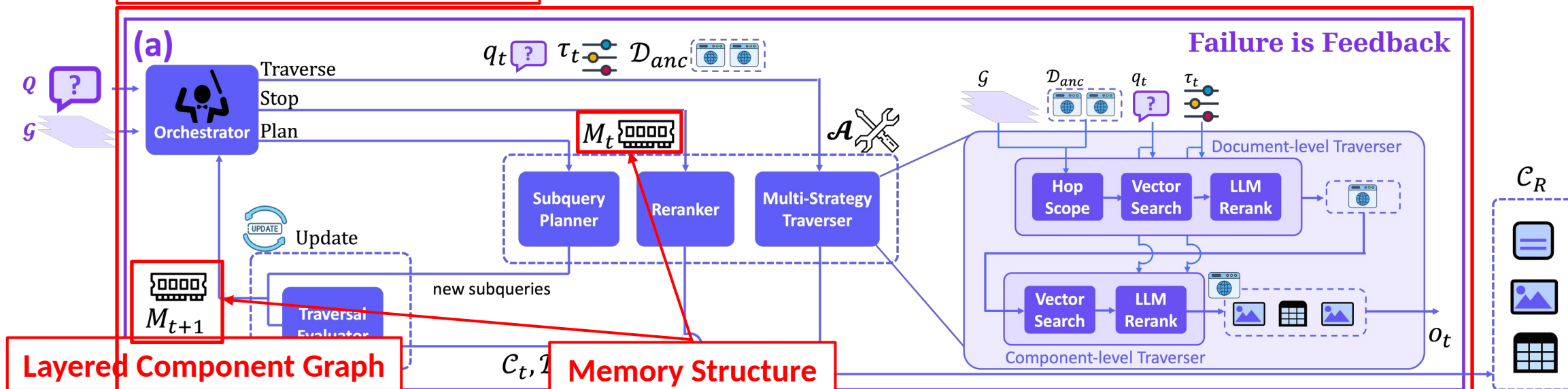
q₂' revised

✓

re-plan

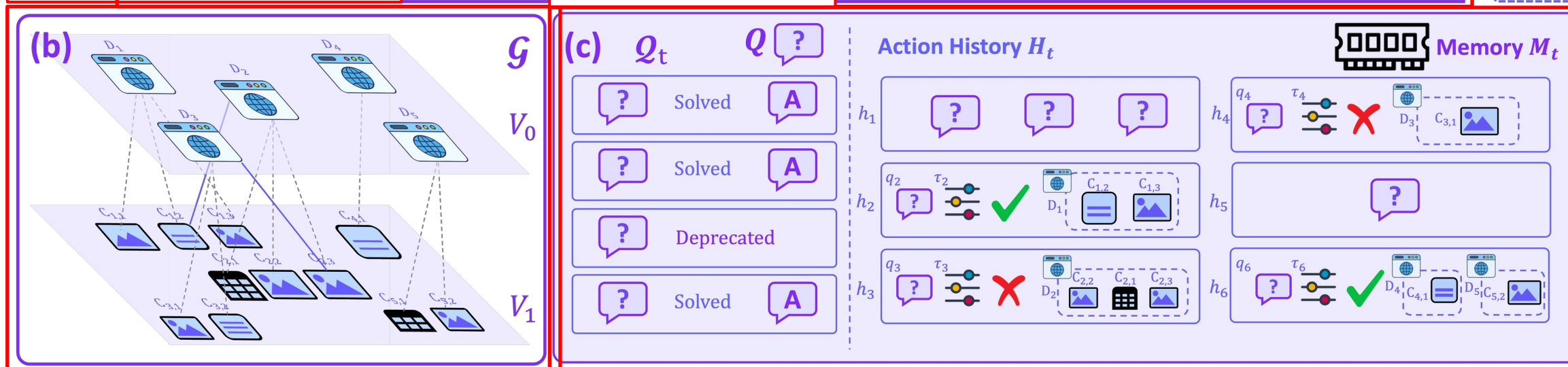
Overview of FiF

Agentic Retrieval Framework



Layered Component Graph

Memory Structure



Running Example

0 PLAN

1 TRAVERSE

2 TRAVERSE

3 TRAVERSE

4 PLAN

+ Traverser

5 TRAVERSE

6 STOP

? How many **words** is the writing behind the statue of the **president who served with Vice President Hannibal Hamlin** ?

ANSWER

24

PLAN COMPLETE Query decomposed into 2 sub-queries.

MEMORY · M t

SUBQUERY LIST · Q t

q₁ who served with Hannibal Hamlin?

PENDING

q₂ words behind him?

PENDING

ACTION HISTORY · H t

h₀ PLAN q₁, q₂ generated

THIS ACTION

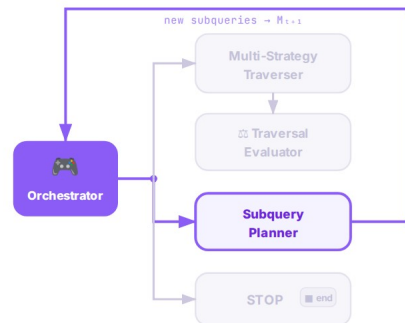
PLAN decompose Q into sub-queries

input original query Q + current sub-query list Q t

output sub-query list Q t

agent Subquery Planner

FIF PIPELINE — ACTIVE NOW

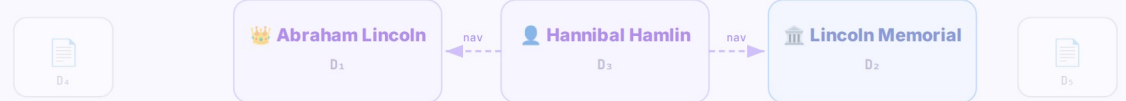


☒ active ☐ idle

GRAPH VIEW · G

$\mathcal{Q}_{anc} = \emptyset$ · GRAPH NOT YET ACCESSED

V₀ · DOCUMENTS



V₁ · COMPONENTS

PLAN runs before any traversal.

The query is decomposed into sub-queries; no anchor is chosen yet ($\mathcal{Q}_{anc} = \emptyset$).

Running Example (Cont.)

0 PLAN 1 TRAVERSE 2 TRAVERSE 3 TRAVERSE 4 PLAN + Traverser 5 TRAVERSE 6 STOP

? How many words is the writing behind the statue of the president who served with Vice President Hannibal Hamlin ?

ANSWER

24

HOP SUCCEEDED · $q_1 \rightarrow \text{Lincoln}$ Global search retrieved a paragraph in the Hamlin document; it resolves q_1 . Logged as h_1 ✓.

MEMORY · M T

SUBQUERY LIST · Q t

q_1 who served with Hannibal Hamlin? ✓ → Lincoln

q_2 words behind him? PENDING

ACTION HISTORY · H t

h_0 PLAN q_1, q_2 generated ✓

h_1 q_1 vec:global Hamlin · $C_{3,1}$ ✓

THIS ACTION

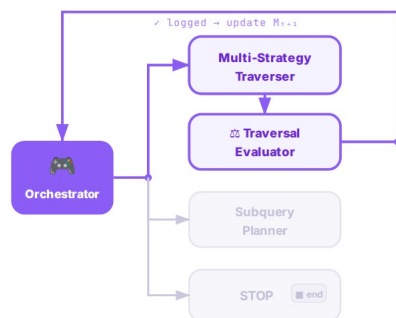
TRAVERSE first hop for q_1 — no anchor yet, search globally

q_t q_1 — "who served with Hannibal Hamlin?"

$\mathcal{D}_{anc}$ empty

τ_t Global hop Vector-only LLM: off

FIF PIPELINE — ACTIVE NOW

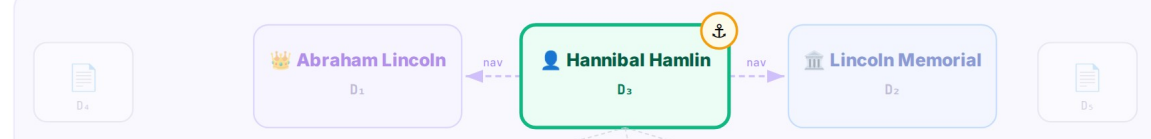


active idle

GRAPH VIEW · G

$\mathcal{D}_{anc} = \{\text{HAMLIN}\}$

V_0 · DOCUMENTS



V_1 · COMPONENTS

¶ PARAGRAPH · $C_{3,1}$

"Hannibal Hamlin served 1861–1865, during President Abraham Lincoln's first term."

IMAGE · $C_{3,2}$



INFOBOX · $C_{3,3}$

15th Vice President
In office: Mar 1861 – Mar 1865
President: **Abraham Lincoln**
Preceded by: J. Breckinridge
Succeeded by: A. Johnson

q_1 resolved → "Abraham Lincoln".

The Hamlin document (D_3) becomes the anchor $\mathcal{D}_{anc}$ for the next hop.

Running Example (Cont.)

0 PLAN

1 TRAVERSE

2 TRAVERSE

3 TRAVERSE

4 PLAN

+ Traverser

5 TRAVERSE

6 STOP

? How many words is the writing behind the statue of the president who served with Vice President Hannibal Hamlin?

ANSWER

24

HOP FAILED · q₂ Reached Lincoln but returned the wrong component — vector match was semantically blind. Logged as h₂ X.

MEMORY · M t

SUBQUERY LIST · Q t

q₁ who served with Hannibal Hamlin? ✓ → Lincoln

q₂ words behind him? X FAIL

ACTION HISTORY · H t

h₀ PLAN q₁, q₂ generated ✓

h₁ q₁ vec:global Hamlin · C_{3,1} ✓

h₂ q₂ vec:local Lincoln · C_{1,4} X

THIS ACTION

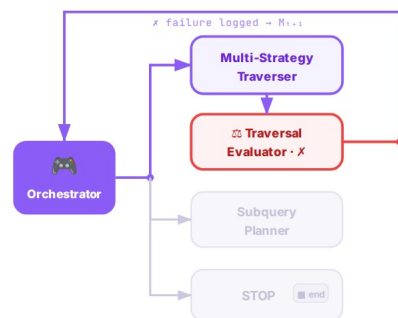
TRAVERSE one hop for q₂ from the anchor

q_t q₂ — "words behind him?"

Q_{anc} {Hamlin}

T_t Local hop Vector-only LLM: off

FIF PIPELINE — ACTIVE NOW



active idle

GRAPH VIEW · G

Q_{anc} = {LINCOLN} · X FAIL

V₀ · DOCUMENTS



V₁ · COMPONENTS

¶ PARAGRAPH · C_{1,4}
"In August 1837, Lincoln wrote Mary Owens a letter half-jokingly proposing marriage..."

▢ INFOBOX · C_{1,2}
Lincoln Cabinet
Office · Name · Term
President: Abraham Lincoln · 1861–1865
Secretary of State: William H. Seward · 1861–1865

X Wrong component. Vector-only matched "writing" ↔ "wrote a letter" — a surface similarity, not the answer.

Anchor moves to Lincoln (D₁); the failure is logged as h₂.

Running Example (Cont.)

0 PLAN 1 TRAVERSE 2 TRAVERSE 3 TRAVERSE 4 PLAN + Traverser 5 TRAVERSE 6 STOP

? How many words is the writing behind the statue of the president who served with Vice President Hannibal Hamlin ?

ANSWER
24

HOP FAILED AGAIN · q₂ A second wrong component inside Lincoln. With no outgoing edge, the search is stuck. Logged as h₃ X.

MEMORY · M T

SUBQUERY LIST · Q t

q₁ who served with Hannibal Hamlin? ✓ → Lincoln

q₂ words behind him? X FAIL x2

ACTION HISTORY · H t

h₀ PLAN q₁, q₂ generated ✓

h₁ q₁ vec·global Hamlin · C_{3,1} ✓

h₂ q₂ vec·local Lincoln · C_{1,4} X

h₃ q₂ vec·local Lincoln · C_{1,2} X

THIS ACTION

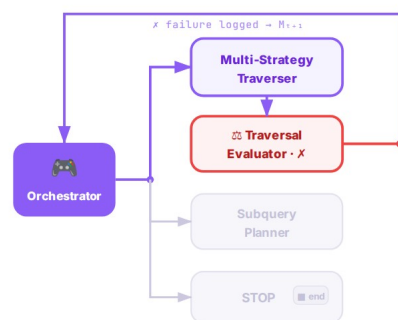
TRAVERSE retry q₂ inside the same anchor (no edge to leave)

q_t q₂ — "words behind him?"

Q_{anc} {Lincoln} (no outgoing edge)

T_t Local hop Vector-only LLM: off

FIF PIPELINE — ACTIVE NOW



active idle

GRAPH VIEW · G

Q_{anc} = {LINCOLN} · X FAIL AGAIN

V₀ · DOCUMENTS



V₁ · COMPONENTS

¶ PARAGRAPH · C_{1,4}
"In August 1837, Lincoln wrote Mary Owens a letter half-jokingly proposing marriage..."

■ INFOBOX · C_{1,2}
Lincoln Cabinet
Office · Name · Term
President: **Abraham Lincoln** · 1861–1865
Secretary of State: William H. Seward · 1861–1865

X Wrong again. The cabinet table is about Lincoln, but says nothing about "words behind a statue".

Two failures inside Lincoln with no way out — the failure trace (h₂, h₃) now signals that re-planning is needed.

Running Example (Cont.)

0 PLAN 1 TRAVERSE 2 TRAVERSE 3 TRAVERSE 4 PLAN + Traverser 5 TRAVERSE 6 STOP

? How many words is the writing behind the statue of the president who served with Vice President Hannibal Hamlin ?

ANSWER
24

+ INTERMISSION Before the retry — a look inside the Multi-Strategy Traverser: the tool q_2' is about to use.

MEMORY · M t

SUBQUERY LIST · Q t

q_1 who served with Hannibal Hamlin? ✓ → Lincoln

q_2 words behind him? DEPRECATED

q_2' words behind statue of Lincoln? NEXT

ACTION HISTORY · H t

h_0 PLAN q_1, q_2 generated ✓

h_1 q_1 vec·global Hamlin · $C_{3,1}$ ✓

h_2 q_2 vec·local Lincoln · $C_{1,4}$ ✗

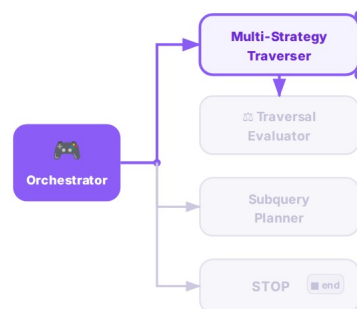
h_3 q_2 vec·local Lincoln · $C_{1,2}$ ✗

h_4 PLAN $q_2 \rightarrow q_2'$ re-planned ✓

THIS ACTION

PREVIEW the Multi-Strategy Traverser, zoomed in
two tiers Document-level → Component-level
per tier Vector Search → optional LLM Rerank
escalate turn on LLM Rerank when a cheap hop fails

FIF PIPELINE — ZOOMING INTO TRAVERSER



☒ active ☐ idle

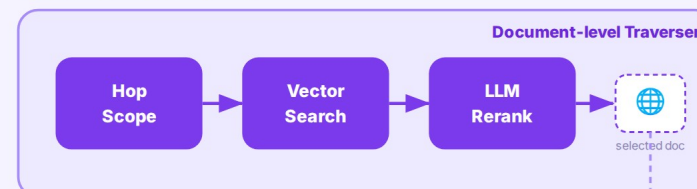
GRAPH VIEW · G

TRAVERSER INTERNALS (ZOOM-IN)

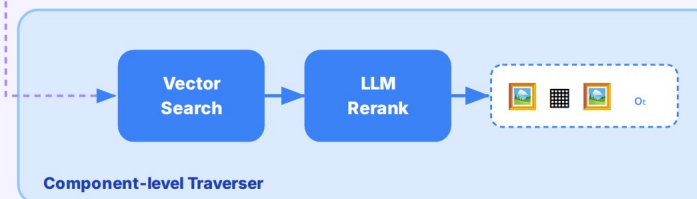
MULTI-STRATEGY TRAVERSER

$\tau = (\text{hop scope, vector granularity, LLM reasoning})$

inputs: $\mathcal{G} \cdot \mathcal{Q}_{anc} \cdot q_t \cdot \tau_t$



selected doc → its components become the next search space



LLM Rerank in each tier is the escalation knob — off by default, on after a failed hop.

Running Example (Cont.)

0 PLAN 1 TRAVERSE 2 TRAVERSE 3 TRAVERSE 4 PLAN + Traverser 5 TRAVERSE 6 STOP

? How many words is the writing behind the statue of the president who served with Vice President Hannibal Hamlin ?

ANSWER
24

HOP SUCCEEDED · $q_2' \rightarrow \text{image}$ LLM rerank found the **Memorial image** ($C_{2,1}$). Every sub-query is now solved. Logged as h_5 ✓.

MEMORY · M T

SUBQUERY LIST · Q t

q_1 who served with Hannibal Hamlin? ✓ → Lincoln

q_2 words behind him? DEPRECATED

q_2' words behind statue of Lincoln? ✓ → image

ACTION HISTORY · H t

h_0 PLAN q_1, q_2 generated ✓

h_1 q_1 vec·global Hamlin · $C_{3,1}$ ✓

h_2 q_2 vec·local Lincoln · $C_{1,4}$ ✗

h_3 q_2 vec·local Lincoln · $C_{1,2}$ ✗

h_4 PLAN $q_2 \rightarrow q_2'$ re-planned ✓

h_5 q_2' llm·local Memorial · $C_{2,1}$ ✓

THIS ACTION

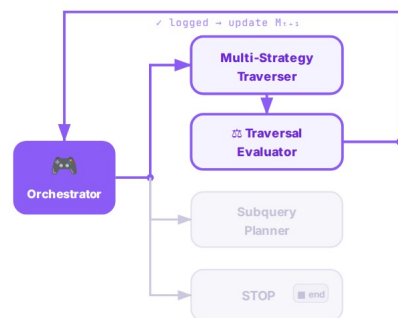
TRAVERSE escalated retry for q_2' — backtrack, then hop

q_t $q_2' \rightarrow$ "words behind statue of Lincoln?"

$\mathcal{D}_{anc}$ {Hamlin} (reset by backtrack)

τ_t Local hop g=2 fine LLM: rerank

FIF PIPELINE — ACTIVE NOW

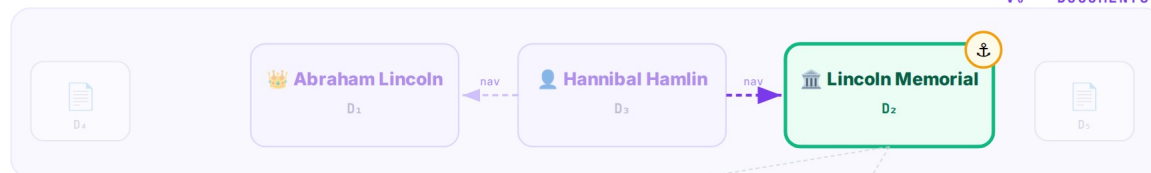


active idle

GRAPH VIEW · G

$\mathcal{D}_{anc} = \{\text{MEMORIAL}\}$ ✓

V_0 · DOCUMENTS



V_1 · COMPONENTS



¶ PARAGRAPH · $C_{2,2}$
Engraving transcript: the Gettysburg Address — its closing line has 24 words.

✓ q_2' resolved → the **Memorial image** ($C_{2,1}$): the engraving behind Lincoln's statue.

Anchor moves to Memorial (D_2); the successful hop is logged as h_5 .

Running Example (Cont.)

0 PLAN 1 TRAVERSE 2 TRAVERSE 3 TRAVERSE 4 PLAN + Traverser 5 TRAVERSE 6 STOP

? How many words is the writing behind the statue of the president who served with Vice President Hannibal Hamlin ?

ANSWER
24

STOP All sub-queries solved — the Orchestrator terminates and the **Reranker** emits the final ranked components $\mathcal{C}_R$.

MEMORY · M T

SUBQUERY LIST · Q t

q₁ who served with Hannibal Hamlin? ✓ → Lincoln

q₂ words behind him? DEPRECATED

q₂' words behind statue of Lincoln? ✓ → image

ACTION HISTORY · H t

h₀ PLAN q₁, q₂ generated ✓

h₁ q₁ vec·global Hamlin · C_{3,1} ✓

h₂ q₂ vec·local Lincoln · C_{1,4} ✗

h₃ q₂ vec·local Lincoln · C_{1,2} ✗

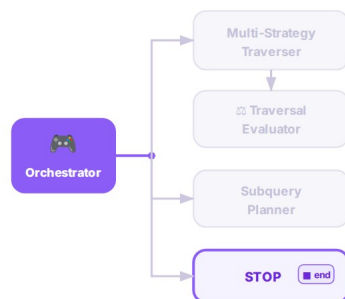
h₄ PLAN q₂ → q₂' re-planned ✓

h₅ q₂' llm·local Memorial · C_{2,1} ✓

THIS ACTION

STOP terminate & assemble the final answer set
trigger all sub-queries solved (q₁ ✓, q₂' ✓)
module Reranker → $\mathcal{C}_R$
answer 24 words (from C_{2,1})

FIF PIPELINE — TERMINAL STEP

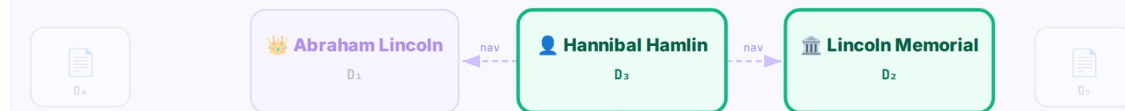


☒ active ☐ idle

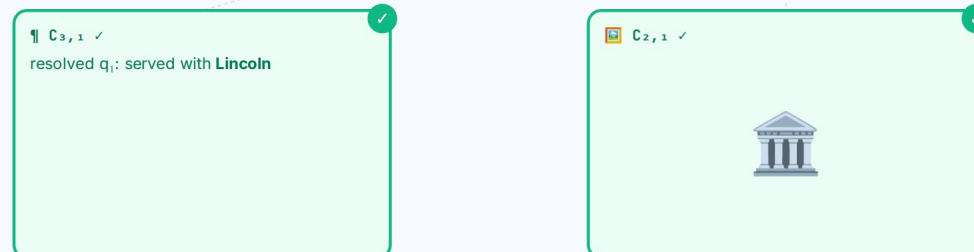
GRAPH VIEW · G

STOP · FINAL $\mathcal{C}_R$

V₀ · DOCUMENTS



V₁ · COMPONENTS

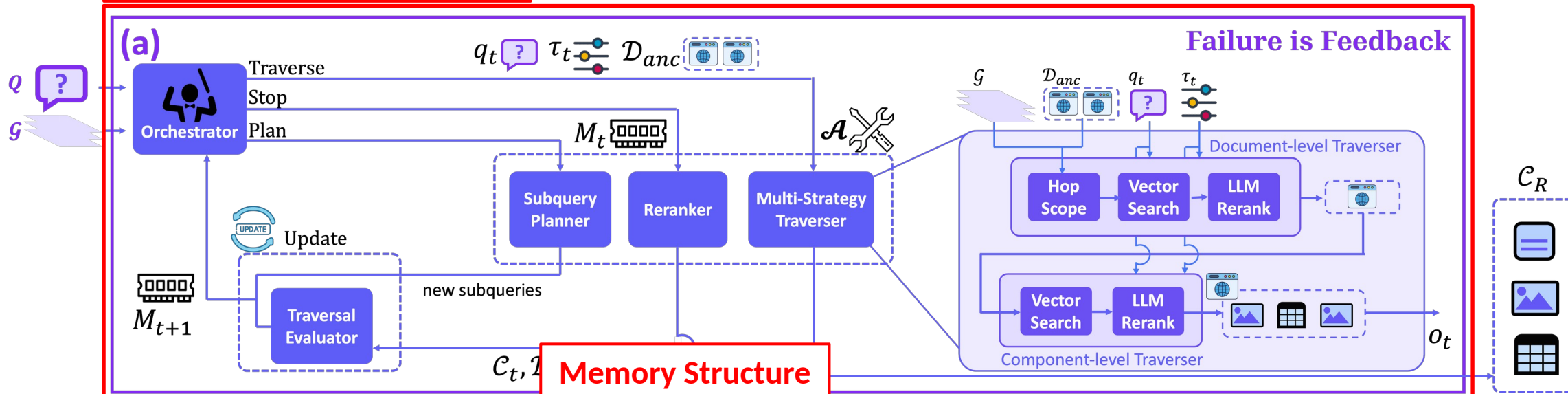
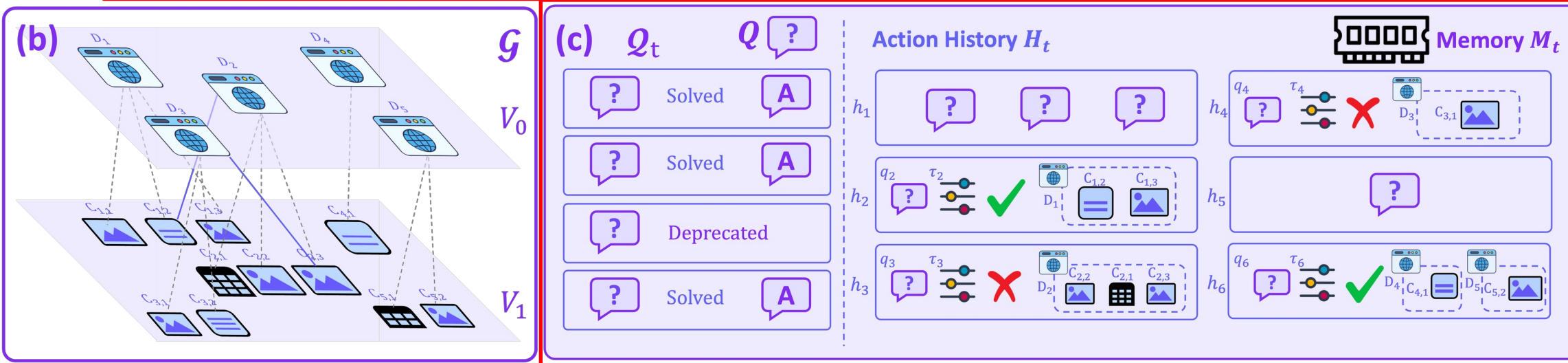


STOP — all sub-queries solved. The Reranker assembles the final ranked components $\mathcal{C}_R$.

Top component (C_{2,1}) yields the answer: the engraving has 24 words.

Overview of FiF (Revisited)

Agentic Retrieval Framework



Retriever Accuracy Comparison

- **FIF** outperforms the previous graph-based SOTA model, **LILaC**, by substantial margins of 22.03% in R@10 and 18.60% in MRR@10, on average.
- Compared to the strongest agentic baselines, **FIF** gains 12.88% / 11.39% in R@10 and 9.31% / 8.92% in MRR@10 over **IRCoT** and **MoLoRAG**, respectively, with improvements particularly strong at low-k (R@1, R@2).

Table 1. Retrieval accuracy (Recall and MRR) of FIF and its competitors on three benchmarks. R@k, M@k indicates recall at k and MRR at k , respectively. The best score in each column is in **bold**, and the second-best is underlined.

Algorithm	MULTIMODALQA ^{Doc}					MMCoQA ^{Doc}					WEBQA ^{Doc}				
	R@1	R@2	R@5	R@10	M@10	R@1	R@2	R@5	R@10	M@10	R@1	R@2	R@5	R@10	M@10
NV-EMBED-V2	26.42	37.63	52.08	61.45	68.13	18.22	28.46	40.22	48.19	41.97	22.18	31.38	39.80	51.19	55.81
VisRAG	34.19	42.12	53.38	56.91	57.88	19.57	27.53	33.59	37.31	30.01	25.58	41.46	46.10	48.75	50.60
COLPALI	38.38	52.61	61.73	63.95	67.65	31.11	41.97	46.89	51.27	43.13	35.21	46.90	53.37	57.78	56.90
LILaC	33.59	50.59	65.13	72.23	79.12	25.25	38.13	51.02	60.86	53.36	32.67	45.35	57.77	64.23	77.87
IRCoT	39.56	55.09	69.87	74.97	82.24	43.03	53.69	62.70	64.62	62.84	40.85	56.73	68.78	73.72	83.15
MoLoRAG	39.91	56.58	70.33	77.82	83.54	43.35	54.55	64.18	67.18	63.18	39.18	54.92	66.18	71.15	82.32
FIF	42.17	57.58	74.88	85.82	86.82	46.31	58.40	69.47	75.17	74.88	43.91	61.78	74.83	80.79	87.77

End-to-End QA Accuracy Comparison

- 16.37% and 16.79% accuracy raise in EM and F1 compared to the previously best graph-based model, **LILaC**.
- 8.71% and 7.14% accuracy raise in EM and F1 compared to the strongest agentic baseline, **IRCoT**.

Table 2. End-to-end QA accuracy (EM and F1) of FiF and its competitors for the three benchmarks. The best score in each column is in **bold**.

Algorithm	MULTIMODALQA ^{Doc}		MMCoQA ^{Doc}		WEBQA ^{Doc}	
	EM	F1	EM	F1	EM	F1
NVEMBED-V2	53.80	61.15	39.20	47.73	51.82	59.88
VISRAG	51.40	61.72	35.20	42.66	46.53	55.19
COLPALI	52.82	63.14	40.73	47.46	49.87	57.78
LILaC	57.78	63.98	42.57	51.14	53.38	60.71
IRCoT	60.32	68.09	47.21	57.71	57.03	65.88
FiF	65.15	70.47	51.11	62.42	62.63	72.47

Algorithm Efficiency

- FIF** runs at slightly lower average wall-clock time than **IRCoT** (116,231 ms vs. 122,458 ms), and is 25.93% faster on WebQA_{Doc} — structure-aware navigation with failure-aware re-anchoring reduces unproductive reasoning.
- Compared to **LILaC**, **FIF** is 5.20–6.71× slower and 9.64–16.60× more expensive — a favorable trade-off given the +22.03% R@10 and +16.37% EM gains.

Table 3. Efficiency (Time, # LLM Calls and API Usage) comparison of FIF and its competitors for the three benchmarks.

Dataset	Algorithm	Time (ms)				# LLM Calls	API Usage		
		Total	LLM	Vector Search	Embedding		# Input Toks	# Output Toks	\$
MULTIMODALQA ^{Doc}	VisRAG	371	0	218	153	0.00	0	0	0.0000
	COLPALI	9,849	0	9,210	639	0.00	0	0	0.0000
	LILaC	19,528	15,943	3,153	432	1.00	1,165	1,119	0.0115
	IRCoT	95,744	95,140	119	484	5.44	41,943	3,296	0.0752
	FIF	114,591	113,642	538	411	7.92	48,863	4,785	0.1109
MMCoQA ^{Doc}	VisRAG	362	0	215	147	0.00	0	0	0.0000
	COLPALI	1,836	0	1,173	663	0.00	0	0	0.0000
	LILaC	20,244	16,879	2,977	388	1.00	1,160	1,041	0.0107
	IRCoT	97,816	97,220	116	479	5.61	52,286	3,360	0.0753
	FIF	105,355	104,630	366	359	7.31	53,149	4,151	0.1037
WEBQA ^{Doc}	VisRAG	386	0	225	161	0.00	0	0	0.0000
	COLPALI	7,919	0	7,298	621	0.00	0	0	0.0000
	LILaC	19,187	14,822	3,782	583	1.00	1,162	737	0.0077
	IRCoT	173,815	173,011	152	651	6.94	65,167	3,913	0.1082
	FIF	128,748	127,664	581	505	8.61	64,683	5,159	0.1278

Conclusion: Two Papers, One Arc

Q1 · HOW TO REPRESENT MULTIMODAL DOCUMENTS

What data structure captures multimodal documents?

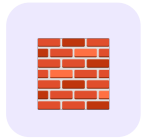
→ **Layered Component Graph** (LILaC, EMNLP'25)

+

Q2 · HOW TO TRAVERSE

How to retrieve effectively over that structure?

→ **Failure-Aware Traversal** (FAT, ICML'26)



01

A graph for multimodal docs

The Layered Component Graph preserves modality and fine granularity, with hierarchical & navigational edges.

representation



02

Failure is feedback

History-aware backtracking re-plans failed sub-queries and escalates scoring strategy cost-effectively.

adaptive traversal



03

Structure + adaptivity = SOTA

Together they set state of the art on open-domain multimodal multi-hop retrieval & QA.

+22.03% R@10 · +16.37% EM

→ **ONGOING**

Reducing today's **inductive bias** by making the unit tools more **fine-grained** — toward retrieval both more **effective** and more **efficient**. We are evolving the data structure further and exploring what new capabilities it should support.

Thank You