MAGE-RAG: MULTIGRANULAR ADAPTIVE GRAPH EVIDENCE FOR AGENTIC MULTIMODAL RAG IN LONG-DOCUMENT QA

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-document multimodal question answering requires evidence whose amount, location, and modality vary with the question. One-shot document RAG typically retrieves a fixed number of text chunks or page images, leaving limited scope to recover missing clues, inspect local details, or remove irrelevant content before answering. Constructing sufficient evidence within a finite budget therefore requires query-dependent decisions about both the extent and granularity of the reader context. We propose MAGE-RAG for budgeted query-time evidence subgraph construction. It connects pages and local elements in an offline graph, uses an online controller to select and revise evidence from retrieved entry pages, and renders the resulting state as page-organized multimodal input. MAGE-RAG achieves 52.75 accuracy on LongDocURL and 53.26 accuracy with 51.19 F1 on MMLongBench-Doc. Breakdown and ablation results support the value of organizing cross-page and heterogeneous evidence; budget experiments further show that increasing the initial page count or controller budget does not consistently improve accuracy. Our code is available at `https://anonymous.4open.science/r/MAGE-RAG-F8B3`.

1 INTRODUCTION

Long-document multimodal question answering draws on evidence in text, tables, charts, and page layouts. Earlier benchmarks and models establish the importance of recognizing and interpreting these forms of content Mathew et al. (2021; 2022); Masry et al. (2022); Xu et al. (2020); Kim et al. (2022). LongDocURL and MMLongBench-Doc extend this setting to long documents, where relevant clues may be sparse and distributed across pages Deng et al. (2025); Ma et al. (2024). The evidence needed to answer a question consequently varies in both scope and form. A local fact may require one region, whereas a comparison can involve several pages and modalities. Answering either type of question requires locating the relevant clues and preserving enough context to interpret them within finite input and computation budgets.

Retrieval-augmented generation (RAG) introduces external evidence before generation Lewis et al. (2020); Gao et al. (2024), and recent systems retrieve OCR chunks or page images for multimodal document QA Faysse et al. (2025); Yu et al. (2025); Cho et al. (2024). Conventional one-shot document RAG nevertheless faces a shared coverage–noise–cost trade-off. **First, fixed Top-k retrieval struggles to balance evidence coverage and context redundancy.** A small k may miss dispersed clues, while a large k can add irrelevant context and input cost. Studies of retrieval granularity and context length document this sensitivity Jiang et al. (2024). **Second, a single retrieval granularity preserves only part of the document evidence.** Text or OCR chunks expose compact local content but may omit layout and visual relations; full-page images retain those relations but also include many unrelated regions Robertson et al. (1995); Khattab et al. (2022); Faysse et al. (2025). **Third, one-shot pipelines provide limited control after retrieval.** They do not explicitly track which clues have been examined, which evidence gaps remain, or which retrieved items should be expanded or removed before answering. These limitations call for evidence construction that adapts both its content and scale to the current query.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

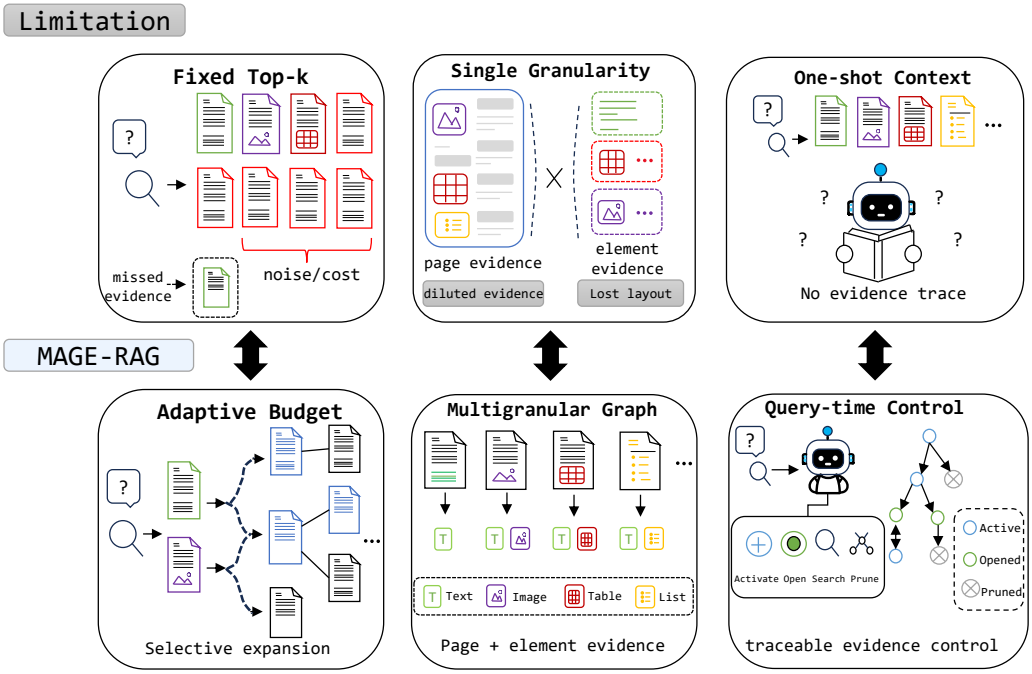

Figure 1: Motivating comparison between existing document RAG methods and MAGE-RAG. The comparison covers three limitations of one-shot retrieval. A fixed Top-k can leave evidence missing or include unnecessary content, a single granularity limits the balance between page context and local detail, and a fixed context prevents further evidence adjustment before answering. MAGE-RAG combines a multigranular page–element graph with query-time control to adapt the selected evidence within execution budgets.

Addressing these limitations requires a query-specific evidence set whose content and scope can be adjusted within a finite budget. Because local clues often depend on their page context, this set must also preserve the relationships between pages and elements. A document graph represents these relationships through containment, reading order, layout, section, and semantic links. Evidence selected from the graph forms a query-specific subgraph, with nodes identifying the retained content and edges preserving its document relationships. Building on structured and active retrieval Yao et al. (2023); Jiang et al. (2023); Asai et al. (2024); Li et al. (2024); He et al. (2024); Peng et al. (2026); Zhang & Wu (2026); Du et al. (2026), we study this process as **budgeted query-time evidence subgraph construction**.

MAGE-RAG (Multigranular Adaptive Graph Evidence for Retrieval-Augmented Generation) constructs this subgraph over an offline graph of full pages and local elements. At query time, visual retrieval supplies initial pages. An online controller then uses the available evidence and action history to select elements for activation or detailed inspection, search for additional pages, and prune evidence. Once evidence construction ends, a structured renderer assembles the retained page context, element summaries, and visual evidence into multimodal input for answer generation.

Contributions. **(1) Formulation.** We formulate evidence organization for long-document multimodal QA as budgeted query-time subgraph construction. **(2) Method.** MAGE-RAG implements this formulation through a shared page–element state, query-time control, and multimodal rendering. **(3) Evaluation.** Experiments on LongDocURL and MMLongBench-Doc assess overall performance, variation across evidence needs, and sensitivity to graph structure and budgets; recorded trajectories support inspection of individual executions.

2 Related Work

Long-document multimodal question answering. Early visual document QA benchmarks cover scanned documents, infographics, and charts through DocVQA, InfographicVQA, and ChartQA Mathew et al. (2021; 2022); Masry et al. (2022). LayoutLM and Donut model document content through text–layout pre-training and OCR-free generation Xu et al. (2020); Kim et al. (2022). Extending evaluation to longer documents, MMLongBench-Doc includes evidence-source annotations, cross-page questions, and unanswerable cases, while LongDocURL tests document understanding, numerical reasoning, and cross-element localization Ma et al. (2024); Deng et al. (2025). These tasks require locating relevant content across a document before integrating it for an answer, motivating retrieval as part of the document QA pipeline.

Document retrieval-augmented generation. Document retrieval methods differ in how much local content and visual context each retrieved unit retains. OCR-based retrieval with BM25 or ColBERTv2 provides compact text chunks Robertson et al. (1995); Khattab et al. (2022), while ColPali retrieves page images through multi-vector visual representations Faysse et al. (2025). VisRAG, M3DocRAG, VDocRAG, and EVisRAG use visual retrieval for document QA and multi-image reasoning Yu et al. (2025); Cho et al. (2024); Tanaka et al. (2025); Sun et al. (2025). Whole pages retain layout but can include irrelevant regions, motivating region-level and on-demand processing in AgenticOCR and RegionRAG Wang et al. (2026); Li et al. (2026). Together with studies of retrieval granularity and context length Gao et al. (2024); Jiang et al. (2024), these methods show the importance of choosing an appropriate evidence unit. The further question of how to revise the selected evidence during answering leads to dynamic retrieval and structured exploration.

Structured and agentic evidence organization. Dynamic retrieval addresses when to seek more information, while graph representations provide relationships along which evidence can be explored. ReAct, FLARE, and Self-RAG study action-based interaction, active retrieval, and self-reflection Yao et al. (2023); Jiang et al. (2023); Asai et al. (2024); GraphReader and G-Retriever study progressive exploration or subgraph selection for long-text and graph QA Li et al. (2024); He et al. (2024); Peng et al. (2026). For multimodal documents, MLDocRAG, LAD-RAG, and G^2-Reader use semantic nodes, layout-aware graphs, or content and planning graphs Zhang & Wu (2026); Sourati et al. (2026); Du et al. (2026), and FlowReader models evidence assembly through minimum-cost flow Mehrish & Vascon (2026). MAGE-RAG brings page and element evidence into a shared query-specific state. Its controller updates that state under execution limits, allowing retained page context and local evidence to be rendered together for the reader.

3 Method

Given a PDF document D and a question q, MAGE-RAG constructs a query-specific evidence subgraph for answer generation. The offline graph links the available pages and elements, the controller selects and revises evidence, and the renderer turns the final state into multimodal reader input (Fig. 2). The following sections describe the evidence state, the relations that support its expansion, and the process that produces the reader context.

3.1 Problem Setup and Overview

We represent the document as $\mathcal{G}_D = (\mathcal{V}^p \cup \mathcal{V}^e, \mathcal{E})$, where $\mathcal{V}^p$ contains page nodes and $\mathcal{V}^e$ contains element nodes. Page nodes retain full-page context; element nodes represent local text, tables, charts, images, and other parser blocks. The edge set $\mathcal{E}$ records relationships along which evidence can be explored.

For each query, the controller maintains a state S_t that records the status $s_t(v)$ of every node at round t. Inactive nodes have not entered the working state. Active nodes provide compact previews, while opened non-page elements also expose detailed content to the controller. Pruned nodes are excluded from the current context, and their operation history remains in the trace. These statuses distinguish the evidence available for further inspection from the content already examined in detail.

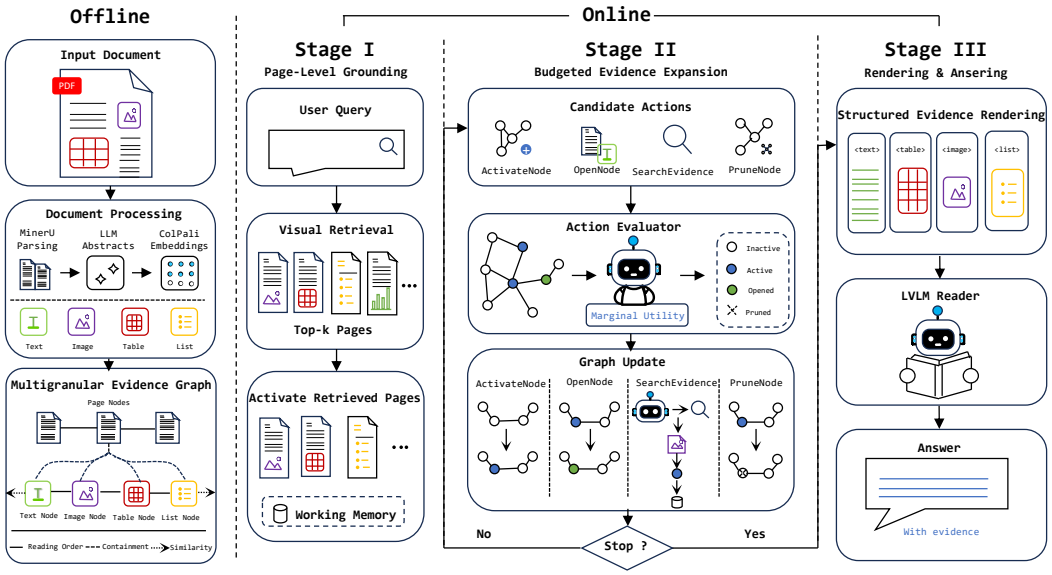

Figure 2: **Overall framework of MAGE-RAG.** An offline evidence graph connects page nodes with fine-grained document elements. At query time, page-level visual retrieval supplies entry pages, an online controller expands or removes evidence, and a structured renderer converts the final state into multimodal reader input.

The active and opened nodes form the working evidence subgraph,

$$\begin{aligned} \mathcal{V}^q_{t,\text{work}} &= \{v \in \mathcal{V}^p \cup \mathcal{V}^e \mid s_t(v) \in \{\mathsf{Active}, \mathsf{Opened}\}\}, \\ \mathcal{G}^q_{t,\text{work}} &= \mathcal{G}_D[\mathcal{V}^q_{t,\text{work}}]. \end{aligned} \tag{1}$$

Here, $\mathcal{G}_D[U]$ denotes the subgraph induced by node set U. Updating the node states changes the retained evidence while preserving its relationships in the document graph. Rendering additionally excludes elements whose parent page is pruned.

The aim is to gather answer-supporting evidence while limiting irrelevant content and reader-input cost. The evaluator guides evidence selection, and execution limits bound the construction process. The final state $S^\star$ is then rendered into the reader input X_{read}. These are design goals implemented through the controller and its budgets; they do not define a separate training objective.

The offline graph is shared across questions about the same document. Each question maintains its own node states and action history, so activating or pruning evidence affects only that query's working context. This separation allows the controller to revisit an element without rebuilding the document representation and keeps the source relationships available throughout evidence construction.

3.2 Multigranular Evidence Graph

Page and element nodes support inspection at different scales. A page preserves the layout and surrounding context of a clue, while an element allows the controller to inspect a particular paragraph, table, chart, or image. A shared page index links each element to its source page, so local evidence can be interpreted in its original context.

Five relation families guide expansion through the graph. Containment connects pages to their elements, and reading order connects consecutive pages and parser-ordered elements. Layout links nearby, horizontally separated elements with vertical overlap; section hierarchy links headings to subsections and content within a page. Semantic relations connect high-scoring neighbors within each node layer. These relations provide routes for local inspection, sequential reading, and exploration of related content elsewhere in the document. Appendix B specifies their construction and traversal directions.

3.3 QUERY-TIME EVIDENCE CONSTRUCTION

Page-level visual retrieval activates the Top-k pages within the allowed document scope to form the initial state S_0. Each subsequent round draws candidates from inactive or pruned elements on active pages and from reachable neighbors of active elements. Candidates are excluded if they lie outside the allowed scope, belong to pruned pages, or are already active or opened. Focused search can introduce additional pages, whose elements become available in the next round.

Search and local inspection operate at complementary scales. A search request activates retrieved pages, after which candidate generation exposes their elements for selection. Opening a selected element gives the evaluator access to fuller content for subsequent decisions. The trace records these changes in sequence, preserving how the current evidence state was reached.

The controller repeats candidate generation, evaluation, and execution for rounds $t = 1, \ldots, T$,

$$\begin{aligned} \mathcal{C}_t &= \mathrm{Candidates}(\mathcal{G}_D, S_{t-1}), \\ d_t &= \mathrm{Eval}_\theta(q, S_{t-1}, \mathcal{C}_t, \tau_{t-1}), \\ (S_t, \tau_t) &= \mathrm{Execute}_B(\mathcal{G}_D, S_{t-1}, \tau_{t-1}, d_t). \end{aligned} \tag{2}$$

Here, $\mathcal{C}_t$ is the candidate list, τ_{t-1} the recorded action history, and d_t the decision returned by the evaluator with parameters θ; B denotes the execution limits described below. The evaluator receives page-organized previews, opened details, a bounded recent trace, and numbered activation candidates. It selects ACTIVATENODE indices and may also request OPENNODE, SEARCHEVIDENCE, or PRUNENODE. The dispatcher $\mathrm{Execute}_B$ validates and applies these requests, records their outcomes, and checks the stopping rules. Normal termination requires a stop flag with no accompanying action requests.

Execution is bounded by the initial page count k, the per-round activation-selection limit a, and the round limit B_T. A cumulative threshold $A_{\max}$ counts attempted activation selections, including invalid indices. This threshold is checked after each activation batch, so the count can reach $A_{\max} + a - 1$. Open, search, and prune requests do not consume that counter. Invalid evaluator output and repeated no-progress rounds also terminate the loop. After termination, a final pass opens up to B_o relevant elements to produce $S^\star$ for rendering. Appendix A specifies the execution order and stopping conditions.

3.4 STRUCTURED MULTIMODAL RENDERING

The renderer turns the final state into page-organized evidence for the reader. It prioritizes explicit question-scope pages when present, then follows page-activation order. Each retained page groups its opened and active elements with their types, abstracts, and available bounding boxes. Full-page images supply global context, while local images or page crops show the corresponding elements. Pruned pages and elements are excluded, and each selected element remains associated with its source page.

The reader receives the question, evidence XML, and interleaved images. Its instructions require answers grounded in the supplied evidence, give precedence to visible content when extracted text conflicts with an image, and specify 'Not answerable.' when support is insufficient. The renderer also records the text and image parts in their input order, allowing the behavior analysis in Section 4.4 to inspect what the reader actually received.

4 EXPERIMENTS

We evaluate query-time evidence construction through overall accuracy, performance across evidence groups, sensitivity to graph structure and execution budgets, and recorded controller behavior. The setup below defines the shared evaluation conditions and the separately marked literature result. This distinction establishes the scope of the comparisons before we examine where MAGE-RAG helps and what costs remain unmeasured.

Table 1: Main results on LongDocURL and MMLongBench-Doc. Values are percentages. Type abbreviations D, T, V, and G/A denote Direct MLLM, Text RAG, Page-level Visual RAG, and Graph/Agentic RAG. † marks results reported by the original paper; other rows use our unified evaluation setting.

(a) LongDocURL

		Evidence Source				Evidence Page			Overall
Method	Type	TXT	LAY	FIG	TAB	SP	MP	CE	Acc
Direct MLLM	D	57.49	41.74	50.21	47.29	51.91	48.44	48.32	50.07
BM25	T	44.74	32.54	37.68	30.71	38.24	34.57	37.84	36.26
ColBERTv2	T	41.81	29.71	32.46	22.67	30.08	33.27	32.52	31.74
M3DocRAG	V	56.20	43.04	50.49	45.62	53.47	45.77	48.51	49.35
EVisRAG	V	48.56	34.21	41.10	41.41	47.36	37.01	42.09	41.84
G^2-Reader	G/A	57.82	38.38	46.69	50.95	54.31	47.44	45.24	50.67
MLDocRAG†	G/A	**65.10**	39.40	48.30	41.10	**66.90**	**56.30**	23.40	50.80
MAGE-RAG	G/A	59.82	**45.08**	**51.00**	**51.60**	58.44	47.69	**51.95**	**52.75**

(b) MMLongBench-Doc

		Evidence Source					Evidence Page			Overall	
Method	Type	TXT	LAY	CHA	TAB	FIG	SIN	MUL	UNA	Acc	F1
Direct MLLM	D	**48.98**	**50.34**	38.01	53.67	41.39	54.83	**38.58**	27.62	43.05	43.63
BM25	T	20.84	14.73	12.36	22.38	3.75	21.74	8.82	85.66	31.16	22.03
ColBERTv2	T	21.26	15.83	11.57	16.04	4.10	18.70	9.70	**87.30**	30.56	20.43
M3DocRAG	V	35.48	35.62	37.32	38.45	39.25	50.40	24.87	34.29	38.21	37.52
EVisRAG	V	37.25	32.91	41.83	41.54	35.23	52.03	24.51	50.00	42.33	40.65
G^2-Reader	G/A	40.99	32.61	41.48	**57.30**	31.25	53.91	30.69	58.87	46.96	45.29
MLDocRAG†	G/A	47.20	37.80	42.70	41.30	31.90	52.60	26.40	71.50	47.90	–
MAGE-RAG	G/A	46.02	46.70	**45.41**	51.17	**42.46**	**60.22**	34.70	68.03	**53.26**	**51.19**

TXT, LAY, CHA, FIG, and TAB denote text, layout, chart, figure, and table evidence. SP/SIN, MP/MUL, CE, and UNA denote single-page, multi-page, cross-element, and unanswerable subsets.

4.1 Experimental Setup

Datasets. We evaluate on LongDocURL and MMLongBench-Doc. LongDocURL contains 2,325 questions covering text, tables, figures, layout, and cross-page evidence Deng et al. (2025). MMLongBench-Doc contains expert-annotated questions over long PDFs, including cross-page, multimodal, and unanswerable cases Ma et al. (2024). Together, the benchmarks test sparse evidence localization, heterogeneous evidence integration, and evidence that is distributed across pages.

Baselines. We group the compared systems into four families. Direct MLLM sends the permitted pages directly to the reader. Text RAG includes BM25 Robertson et al. (1995) and ColBERTv2 Khattab et al. (2022), which retrieve OCR text chunks. Page-level Visual RAG includes M3DocRAG Cho et al. (2024), VisRAG Yu et al. (2025), and EVisRAG/VisRAG 2.0 Sun et al. (2025). Graph/Agentic RAG includes MLDocRAG Zhang & Wu (2026) and G^2-Reader Du et al. (2026). These groups provide comparisons between text-only retrieval, whole-page visual retrieval, and structured or multi-step evidence exploration.

Protocol. Our reproduced systems use Qwen3-VL-8B-Instruct as the reader Bai et al. (2025); the MLDocRAG row is taken from its paper and is marked separately in Table 1. MAGE-RAG also uses Qwen3-VL-8B-Instruct as its action evaluator. For retrieval-based methods, the retrieval space is restricted to the document or pages supplied by each benchmark. We report accuracy on LongDocURL and both accuracy and F1 on MMLongBench-Doc. We also report benchmark-provided breakdowns by evidence source, page range, answerability, document type, and answer format. For MAGE-RAG, the controller records activated pages, selected nodes, search and prune actions, and stopping decisions; these traces support the case analysis in Appendix F.

4.2 Main Results

Table 1 reports the overall results. MAGE-RAG obtains 52.75 accuracy on LongDocURL and 53.26 accuracy with 51.19 F1 on MMLongBench-Doc. On the latter benchmark, G^2-Reader obtains 46.96

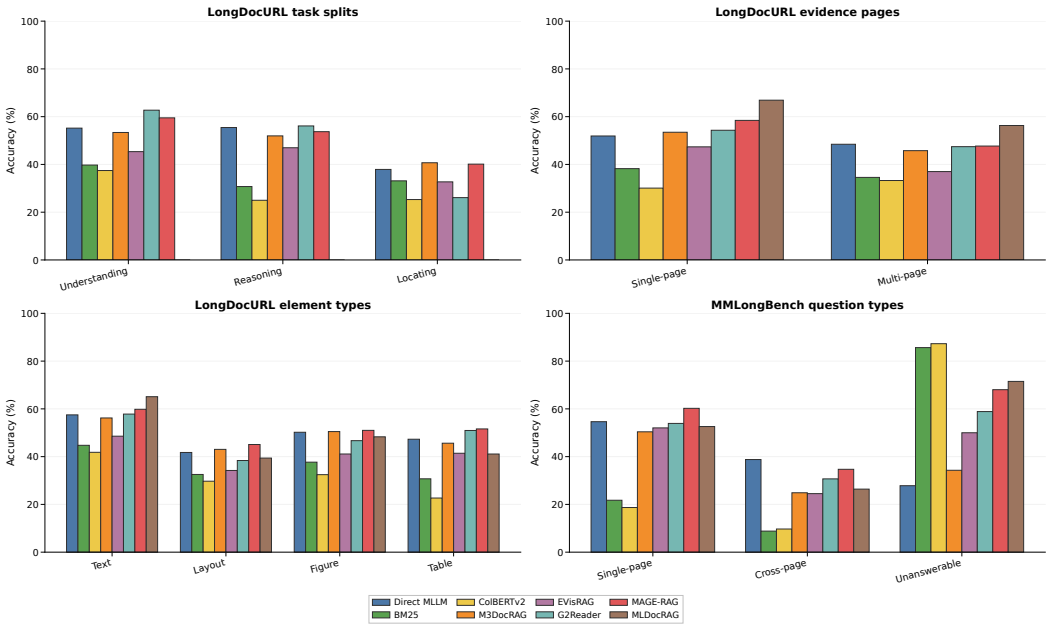

Figure 3: Fine-grained breakdown results on LongDocURL and MMLongBench-Doc. LongDocURL is split by task type, number of evidence pages, and element type; MMLongBench-Doc is split by single-page, cross-page, and unanswerable questions.

accuracy and 45.29 F1. These aggregate differences establish the outcome of the evaluated pipelines. To understand which evidence needs are associated with the gains, we next examine the benchmark breakdowns; component effects are assessed separately through ablations.

The breakdowns in Fig. 3 clarify where the aggregate gains arise. On LongDocURL, MAGE-RAG performs strongly on understanding, reasoning, single-page, and several visual evidence groups, while its multi-page result remains close to Direct MLLM and its locating result remains close to M3DocRAG. On MMLongBench-Doc, MAGE-RAG reaches 60.22 on the single-page group and 34.70 on the cross-page group; the latter is higher than M3DocRAG (24.87) and EVisRAG (24.51). Text baselines obtain higher scores on the unanswerable subset, but their overall accuracy and F1 are substantially lower. This difference cautions against interpreting unanswerable accuracy in isolation and suggests that the main advantage of MAGE-RAG lies in answerable questions requiring visual or cross-page evidence organization.

The subgroup results also identify conditions where other methods remain competitive. MLDocRAG reports the strongest LongDocURL text and single-page scores in the table, under its original evaluation setting, and Direct MLLM is competitive on some page-level groups. The evidence therefore supports an overall advantage in the evaluated setting with variation across evidence needs. Budget sweeps and ablations examine which design choices contribute to this behavior.

4.3 Budget and Ablation Analysis

Budget sensitivity. We vary initial pages k, the round limit T (denoted B_T in the method), and numbered selections per round a. $T = 0$ denotes initial page grounding followed by rendering without online controller iterations. Table 2 and Fig. 4 show that the relationship between budget and accuracy is non-monotonic. With $T = 5$ and $a = 3$, increasing k from 1 to 5 raises accuracy from 48.95 to 53.05 and cross-page accuracy from 24.42 to 34.42; increasing k to 8 lowers both metrics. At $k = 10$, cross-page accuracy rises to 37.55, while overall accuracy remains lower at 50.10, showing that question groups respond differently to the same budget. Across controller settings, moving from $T = 0$ to $T = 5$ generally improves cross-page performance, while the $T = 10$ result with $k = 3, a = 3$ does not improve over the strongest $T = 5$ settings. The observed non-monotonicity supports selecting budgets according to evidence needs. Redundancy

Table 2: Budget sensitivity of MAGE-RAG on MMLongBench-Doc. All runs use the full controller and full evidence graph; values are percentages. Panels (a)–(c) vary one budget while fixing the other two; their shared reference configuration $(k, T, a) = (3, 5, 3)$ is repeated for readability. Panel (d) contains the remaining joint configurations.

(a) Initial pages: $T = 5, a = 3$

k	1	3	5	8	10
Acc	48.95	52.25	53.05	51.53	50.10
F1	45.75	49.44	51.12	49.34	58.61
Single	55.87	59.43	60.16	61.41	57.97
Cross	24.42	31.10	34.42	30.36	37.55
Unans.	73.77	70.90	67.62	65.16	62.29

(b) Controller rounds: $k = 3, a = 3$

T	0	1	3	5	10
Acc	50.01	49.68	50.68	52.25	52.04
F1	46.82	46.75	48.16	49.44	49.45
Single	55.30	56.76	58.27	59.43	59.11
Cross	25.86	25.41	28.80	31.10	30.67
Unans.	76.64	73.36	69.67	70.90	71.72

(c) Actions per round: $k = 3, T = 5$

a	1	3	5	8	10
Acc	52.97	52.25	51.35	51.32	53.14
F1	49.80	49.44	48.94	48.34	50.02
Single	61.79	59.43	59.32	58.77	59.55
Cross	29.87	31.10	29.97	28.41	32.07
Unans.	71.72	70.90	69.26	72.13	72.93

(d) Additional joint configurations

(k, T, a)	Acc	F1	Single	Cross	Unans.
$(3, 0, 5)$	49.77	46.98	56.52	24.35	75.82
$(3, 1, 5)$	48.85	47.04	56.83	26.09	68.03
$(3, 3, 5)$	51.29	49.08	60.82	26.83	69.90
$(5, 10, 5)$	53.26	51.19	60.22	34.70	68.03

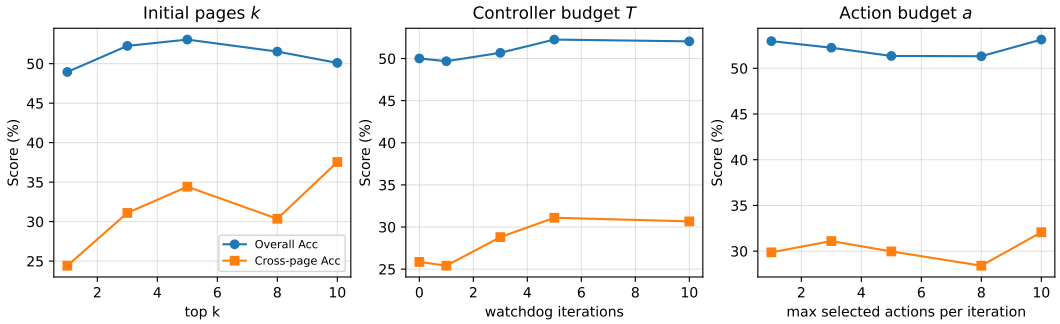

Figure 4: Budget sensitivity on MMLongBench-Doc. The curves vary initial pages k, controller rounds T, and actions per round a.

and irrelevant context are possible explanations for declining accuracy, but these scores alone do not identify the cause.

Ablations. The ablations in Table 3 separate the effects of local rendering, graph expansion, online search, and graph structure. Adding node rendering to top-k page retrieval improves accuracy from 48.75 to 49.71 and F1 from 45.71 to 47.38. Graph neighbor expansion reaches 52.88 accuracy and 50.20 F1 in the reported batch, showing competitive performance in that batch. The no-search variant obtains 22.15 cross-page accuracy, compared with 34.70 for the full configuration in the main results. Among the restricted graph variants, the semantic and page-only graphs share the highest overall accuracy (52.96), while the page-only graph has the highest cross-page accuracy (31.70). The variants therefore expose different trade-offs and do not support attributing the full result to a single relation type.

4.4 Behavior and Limitations

The recorded trajectory links the preceding results to inspectable execution behavior. For each question, it shows retrieved pages, attempted state changes, focused searches, stopping decisions, and the final reader input. These records help identify where to inspect a failure, from missed initial evidence to an inadequate final context. Appendix E reports similar average action counts for correct and incorrect samples in a representative run, and Appendix F illustrates why individual evidence and reader inputs still need inspection. Action counts alone cannot determine the cause of an error.

This diagnostic visibility comes with additional evaluator calls and state-maintenance work, and the method depends on the quality of the offline graph. The current experiments characterize accuracy and logical input quantities, but serving concurrency and backend load are not controlled for an

Table 3: Capability and graph-structure ablations of MAGE-RAG on MMLongBench-Doc. All runs fix $k = 3, T = 10, a = 5$; Acc/F1 are reported as percentages.

Controller	Graph	Acc	F1	Single	Cross	Unans.
Topk Page Only	Full Graph	48.75	45.71	55.30	24.31	73.36
Topk Page With Node Rendering	Full Graph	49.71	47.38	57.09	26.42	71.31
Graph Neighbor Expansion	Full Graph	52.88	50.20	59.83	32.11	76.23
Dynamic Controller No Search	Full Graph	47.95	45.52	55.76	22.15	72.54
Full	Containment Only	51.21	48.17	58.42	28.27	71.72
Full	Layout Graph	50.71	47.98	58.08	29.11	68.85
Full	Page Only	52.84	49.35	58.62	31.70	72.13
Full	Semantic Graph	52.96	49.91	60.37	29.19	75.23
Full	Structural Graph	50.05	47.51	58.69	26.53	69.01
Dynamic Controller No Prune	Full Graph	52.99	50.26	60.64	31.45	70.48

end-to-end latency comparison. The results are most encouraging for questions requiring heterogeneous or cross-page evidence. Whether the additional control is worthwhile for short documents and simple single-hop questions requires cost measurements under a fixed serving environment.

5 Conclusion

MAGE-RAG adapts the content and granularity of long-document QA evidence through query-time construction over a graph of pages and local elements. Results on LongDocURL and MMLongBench-Doc support its overall effectiveness, with benefits that vary across evidence needs and design choices. The budget experiments also show that retrieving more pages or allowing more controller rounds does not consistently improve answers, underscoring the importance of evidence selection within the available budget.

Graph quality and the cost of online control remain important limits on this approach. Evaluation on two benchmarks without a controlled wall-clock comparison leaves broader applicability and efficiency to be established. Better relations among tables, charts, captions, and cross-page sections could improve evidence localization. Manually labeled errors, recorded trajectories, and cost measurements under fixed serving conditions would help determine when the additional control is useful.

AI Use Statement

Generative AI tools, including ChatGPT and OpenAI Codex, were used during this project for code drafting and debugging, literature discovery, discussion of experimental design and result interpretation, and drafting and revising the manuscript. The authors reviewed the resulting code and prose, checked technical descriptions against the implementation and configurations, checked reported values against the available result artifacts, and manually verified cited sources before inclusion. AI tools did not determine authorship or replace the authors' responsibility for the research design, experiments, interpretation, claims, and final manuscript.

Reproducibility Statement

The main paper specifies the method, evaluation setting, primary metrics, and principal results. Appendices A–E provide the state and action definitions, query-time algorithm, graph construction, controller and rendering details, experimental protocol, and diagnostic trace statistics. The anonymous repository at `https://anonymous.4open.science/r/MAGE-RAG-F8B3` provides the implementation, Hydra configurations, benchmark adapters, preprocessing and graph-building scripts, artifact verification utilities, and commands for running evaluation. Dataset files and model weights are obtained from their respective providers and are not redistributed as part of the paper source.

REFERENCES

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/25f7be9694d7b32d5cc670927b8091e1-Abstract-Conference.html.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-VL technical report, 2025. URL https://arxiv.org/abs/2511.21631.

Jaemin Cho, Debanjan Mahata, Ozan Irsoy, Yujie He, and Mohit Bansal. M3DocRAG: Multi-modal retrieval is what you need for multi-page multi-document understanding, 2024. URL https://arxiv.org/abs/2411.04952.

Chao Deng, Jiale Yuan, Pi Bu, Peijie Wang, Zhong-Zhi Li, Jian Xu, Xiao-Hui Li, Yuan Gao, Jun Song, Bo Zheng, and Cheng-Lin Liu. LongDocURL: A comprehensive multimodal long document benchmark integrating understanding, reasoning, and locating. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1135–1159, 2025. doi: 10.18653/v1/2025.acl-long.57. URL https://aclanthology.org/2025.acl-long.57/.

Yaxin Du, Junru Song, Yifan Zhou, Cheng Wang, Jiahao Gu, Zimeng Chen, Menglan Chen, Wen Yao, Yang Yang, Ying Wen, and Siheng Chen. G^2-Reader: Dual evolving graphs for multimodal document QA. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=1NACQKPp1n.

Manuel Faysse, Hugues Sibille, Tony Wu, Bilel Omrani, Gautier Viaud, Céline Hudelot, and Pierre Colombo. ColPali: Efficient document retrieval with vision language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/99e9e141aafc314f76b0ca3dd66898b3-Abstract-Conference.html.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey, 2024. URL https://arxiv.org/abs/2312.10997.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V. Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. G-Retriever: Retrieval-augmented generation for textual graph understanding and question answering. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-4224. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/efaf1c9726648c8ba363a5c927440529-Abstract.html.

Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7969–7992, 2023. doi: 10.18653/v1/2023.emnlp-main.495. URL https://aclanthology.org/2023.emnlp-main.495/.

Ziyan Jiang, Xueguang Ma, and Wenhu Chen. LongRAG: Enhancing retrieval-augmented generation with long-context LLMs, 2024. URL https://arxiv.org/abs/2406.15319.

Omar Khattab, Keshav Santhanam, Xiang Lisa Li, David Hall, Percy Liang, Christopher Potts, and Matei Zaharia. ColBERTv2: Efficient and effective retrieval via lightweight late interaction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 3715–3734, 2022. doi: 10.18653/v1/2022.naacl-main.272. URL https://aclanthology.org/2022.naacl-main.272/.

Geewook Kim, Teakgyu Hong, Moonbin Yim, Jeongyeon Nam, Jinyoung Park, Jinyeong Yim, Wonseok Hwang, Sangdoo Yun, Dongyoon Han, and Seunghyun Park. OCR-free document understanding transformer. In *Computer Vision – ECCV 2022*, volume 13688 of *Lecture Notes in Computer Science*, pp. 498–517, 2022. doi: 10.1007/978-3-031-19815-1_29. URL https://doi.org/10.1007/978-3-031-19815-1_29.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/6b493230-Abstract.html.

Shilong Li, Yancheng He, Hangyu Guo, Xingyuan Bu, Ge Bai, Jie Liu, Jiaheng Liu, Xingwei Qu, Yangguang Li, Wanli Ouyang, Wenbo Su, and Bo Zheng. GraphReader: Building graph-based agent to enhance long-context abilities of large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 12758–12786, 2024. doi: 10.18653/v1/2024.findings-emnlp.746. URL https://aclanthology.org/2024.findings-emnlp.746/.

Yinglu Li, Zhiying Lu, Zhihang Liu, Yiwei Sun, Chuanbin Liu, and Hongtao Xie. RegionRAG: Region-level retrieval-augmented generation for visual document understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40(8), pp. 6662–6670, 2026. doi: 10.1609/aaai.v40i8.37597. URL https://ojs.aaai.org/index.php/AAAI/article/view/37597.

Yubo Ma, Yuhang Zang, Liangyu Chen, Meiqi Chen, Yizhu Jiao, Xinze Li, Xinyuan Lu, Ziyu Liu, Yan Ma, Xiaoyi Dong, Pan Zhang, Liangming Pan, Yu-Gang Jiang, Jiaqi Wang, Yixin Cao, and Aixin Sun. MMLongBench-Doc: Benchmarking long-context document understanding with visualizations. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-3041. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/ae0e43289bffea0c1fa34633fc608e92-Abstract-Datasets_and_Benchmarks_Track.html.

Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. ChartQA: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 2263–2279, 2022. doi: 10.18653/v1/2022.findings-acl.177. URL https://aclanthology.org/2022.findings-acl.177/.

Minesh Mathew, Dimosthenis Karatzas, and C. V. Jawahar. DocVQA: A dataset for VQA on document images. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 2200–2209, 2021. URL https://openaccess.thecvf.com/content/WACV2021/html/Mathew_DocVQA_A_Dataset_for_VQA_on_Document_Images_WACV_2021_paper.html.

Minesh Mathew, Viraj Bagal, Rubèn Pérez Tito, Dimosthenis Karatzas, Ernest Valveny, and C. V. Jawahar. InfographicVQA. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1697–1706, 2022. URL https://openaccess.thecvf.com/content/WACV2022/html/Mathew_InfographicVQA_WACV_2022_paper.html.

Ambuj Mehrish and Sebastiano Vascon. FlowReader: Min-cost flow optimization for multi-modal long document Q&A, 2026. URL https://arxiv.org/abs/2606.07235.

Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. Graph retrieval-augmented generation: A survey. *ACM Transactions on Information*

Systems, 44(2):1–52, 2026. doi: 10.1145/3777378. URL https://doi.org/10.1145/3777378.

Stephen E. Robertson, Steve Walker, Susan Jones, Micheline M. Hancock-Beaulieu, and Mike Gatford. Okapi at TREC-3. *NIST Special Publication*, 500-225:109–126, 1995.

Zhivar Sourati, Zheng Wang, Marianne Menglin Liu, Yazhe Hu, Mengqing Guo, Sujeeth Bharadwaj, Kyu Han, Tao Sheng, Sujith Ravi, Morteza Dehghani, and Dan Roth. LAD-RAG: Layout-aware dynamic RAG for visually-rich document understanding. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15945–15968, 2026. doi: 10.18653/v1/2026.acl-long.724. URL https://aclanthology.org/2026.acl-long.724/.

Yubo Sun, Chunyi Peng, Yukun Yan, Shi Yu, Zhenghao Liu, Chi Chen, Zhiyuan Liu, and Maosong Sun. VisRAG 2.0: Evidence-guided multi-image reasoning in visual retrieval-augmented generation, 2025. URL https://arxiv.org/abs/2510.09733.

Ryota Tanaka, Taichi Iki, Taku Hasegawa, Kyosuke Nishida, Kuniko Saito, and Jun Suzuki. VDocRAG: Retrieval-augmented generation over visually-rich documents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 24827–24837, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Tanaka_VDocRAG_Retrieval-Augmented_Generation_over_Visually-Rich_Documents_CVPR_2025_paper.html.

Zhengren Wang, Dongsheng Ma, Huaping Zhong, Jiayu Li, Wentao Zhang, Bin Wang, and Conghui He. AgenticOCR: Parsing only what you need for efficient retrieval-augmented generation. In *Advances in Neural Information Processing Systems*, volume 39, 2026. URL https://neurips.cc/virtual/2026/poster/153107.

Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. LayoutLM: Pre-training of text and layout for document image understanding. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1192–1200, 2020. doi: 10.1145/3394486.3403172. URL https://doi.org/10.1145/3394486.3403172.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Shi Yu, Chaoyue Tang, Bokai Xu, Junbo Cui, Junhao Ran, Yukun Yan, Zhenghao Liu, Shuo Wang, Xu Han, Zhiyuan Liu, and Maosong Sun. VisRAG: Vision-based retrieval-augmented generation on multi-modality documents. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/3640a1997a4c9571cea9db2c82e1fc35-Abstract-Conference.html.

Yongyue Zhang and Yaxiong Wu. MLDocRAG: Multimodal long-context document retrieval augmented generation, 2026. URL https://arxiv.org/abs/2602.10271.

A Detailed Formulation and Algorithm

This appendix expands the query-time formulation and specifies how the abstract evidence-management process maps to the implemented controller. The main paper retains the definitions needed to understand the method, while the details below distinguish graph-state operations, evaluator requests, termination, and the budgets applied during execution.

A.1 Working Evidence State

Given a document $D = \{P_i\}_{i=1}^{n}$ and a question q, MAGE-RAG constructs an answer from a document evidence graph $\mathcal{G}_D = (\mathcal{V}, \mathcal{E})$. Its node set contains page nodes $\mathcal{V}^p$ and in-page element nodes $\mathcal{V}^e$. During query-time construction, every node has one of four states:

$$s_t(v) \in \{\mathsf{Inactive}, \mathsf{Active}, \mathsf{Opened}, \mathsf{Pruned}\}. \tag{3}$$

An inactive node has not entered the current working state. An active page or element is available to subsequent decisions through its location and compact representation. Opening is defined only for non-page elements and exposes more detailed content, including available text, bounding boxes, and image references. Pruning removes a page or element from the current evidence context while retaining the operation in the trace; it does not delete the node from the offline graph.

The working node set and its induced subgraph are

$$\begin{aligned} \mathcal{V}^q_{t,\mathrm{work}} &= \{v \mid s_t(v) \in \{\mathsf{Active}, \mathsf{Opened}\}\}, \\ \mathcal{G}^q_{t,\mathrm{work}} &= \mathcal{G}_D[\mathcal{V}^q_{t,\mathrm{work}}]. \end{aligned} \tag{4}$$

After query-time construction, the renderer converts the final state into a page-organized multimodal input $X_{\mathrm{read}} = \mathrm{Render}(q, \mathcal{G}_D, S^\star)$. The controller seeks answer-supporting evidence while limiting irrelevant content and reader-input cost, following the execution rules below.

A.2 Operations and Termination

MAGE-RAG uses several related operations, but they enter the execution path in different ways. Initial page retrieval activates the Top-k entry pages through ActivatePage. The candidate generator then exposes numbered ActivateNode candidates from elements on active pages and from graph neighbors of active elements. The evaluator selects among these numbered candidates and may separately request OpenNode, SearchEvidence, or PruneNode.

OpenNode applies only to an active non-page element. SearchEvidence issues one focused query within the allowed pages of the same document; the retrieved pages are activated and expose their elements in the following candidate-generation round. PruneNode changes the state of a visible page or element and records a reason. A pruned element can later be activated again, but elements under a pruned parent page remain unavailable until the page state changes. Termination uses a Boolean stop decision. The online loop stops normally only when stop is returned without any activation, opening, search, or pruning request.

Each successful state operation, evaluator decision, search result, validation error, and stopping reason is recorded in the trace τ. The trace preserves the sequence that produced the final reader context and supports the behavior analyses reported in the paper.

A.3 Execution Budgets

Let k be the initial page count, a the per-round numbered selection limit, $A_{\max}$ the cumulative stopping threshold, B_T the round limit, and B_o the final-open limit. Let m_t count processed selections, including invalid indices, and $n_t = \sum_{i=1}^{t} m_i$. Execution follows

$$\begin{gathered} |\mathcal{P}_0| \le k, \quad m_t \le a, \quad T \le B_T, \quad |\mathcal{V}_{\text{final-open}}| \le B_o, \\ n_t \ge A_{\max} \implies \text{stop after the activation batch.} \end{gathered} \tag{5}$$

Each round first truncates numbered selections to a, processes that batch, and then checks $n_t \ge A_{\max}$. Because the check follows the batch, the cumulative count can reach $A_{\max} + a - 1$. Invalid indices still consume the counter. Open, search, and prune requests are processed only if this

threshold has not ended the loop and do not consume the numbered-selection counter. Round and no-progress limits provide additional stopping conditions. The final pass opens up to B_o relevant elements, including eligible elements on newly activated pages. Reader input size and model-call counts are recorded separately.

A.4 Query-Time Algorithm

Algorithm 1 follows the implemented execution order. The notation $\mathrm{Requests}(d_t)$ denotes all non-stop requests in the evaluator decision, and Take_a retains the first a numbered selections. $\mathrm{ApplyActivate}$ validates those selections and records attempted operations, including invalid indices.

Algorithm 1 MAGE-RAG Query-Time Evidence Construction

Require: Question q, document D, graph $\mathcal{G}_D$, budgets $k, a, A_{\max}, B_T, B_o$
Ensure: Answer y and evidence trace τ

$\mathcal{P}_0 \leftarrow \mathrm{TopKPageGrounding}(q, D, k)$
$S \leftarrow \mathrm{ActivatePages}(\mathcal{P}_0)$, $\tau \leftarrow [\,]$, $n_{\mathrm{act}} \leftarrow 0$
for $t = 1$ to B_T **do**
  $\mathcal{C}_t \leftarrow \mathrm{GenerateCandidates}(\mathcal{G}_D, S)$
  $d_t \leftarrow \mathrm{Eval}_\theta(q, S, \mathcal{C}_t, \tau)$
  if $d_t.\mathrm{stop}$ and $\mathrm{Requests}(d_t) = \varnothing$ **then**
    break
  end if
  $A_t^{\mathrm{act}} \leftarrow \mathrm{Take}_a(d_t.\mathrm{selected_actions})$
  $S \leftarrow \mathrm{ApplyActivate}(S, A_t^{\mathrm{act}})$, $n_{\mathrm{act}} \leftarrow n_{\mathrm{act}} + |A_t^{\mathrm{act}}|$
  if $n_{\mathrm{act}} \geq A_{\max}$ **then**
    break
  end if
  $S \leftarrow \mathrm{ApplyOpen}(S, d_t.\mathrm{open})$
  if $d_t.\mathrm{search} \neq \varnothing$ **then**
    $\mathcal{P}_t \leftarrow \mathrm{PageSearch}(d_t.\mathrm{search}, D, S)$
    $S \leftarrow \mathrm{ActivatePages}(S, \mathcal{P}_t)$
  end if
  $S \leftarrow \mathrm{ApplyPrune}(S, d_t.\mathrm{prune})$
  $\tau \leftarrow \mathrm{UpdateTrace}(\tau, d_t, S)$
  if the no-progress watchdog is reached **then**
    break
  end if
end for
$S^\star \leftarrow \mathrm{FinalOpenActiveEvidence}(S, B_o)$
$X_{\mathrm{read}} \leftarrow \mathrm{Render}(q, \mathcal{G}_D, S^\star)$
$y \leftarrow \mathrm{Reader}(X_{\mathrm{read}})$
return y, τ

A.5 Computational Scope

Let $\mathcal{E}_{\mathrm{struct}}$ contain the structural edges and let each semantic layer retain at most K_s neighbors per node. The stored graph then contains $O(|\mathcal{E}_{\mathrm{struct}}| + |\mathcal{V}|K_s)$ edges under a fixed K_s. Query-time candidate generation is local to active pages and graph neighborhoods; it does not enumerate every graph node in each round. If H_t denotes the visible evidence-state size and C_t the candidate count in round t, the controller processes inputs whose scale depends on $H_t + C_t$ over at most B_T calls. Final reader cost additionally depends on the rendered text parts and images. These quantities describe the scope of the computation, but they do not establish an empirical efficiency advantage over methods whose costs were not measured under the same protocol.

B Evidence Graph Construction

This appendix describes how the offline pipeline converts each PDF into the document evidence graph used at query time. The graph is built once per document from page images, page-level

retrieval embeddings, and structured blocks extracted by the document parser. Its records are stored as document metadata, node records, and edge records; the online controller reads these artifacts without modifying the offline graph.

B.1 Node Records and Element Types

For a document D, the node set is partitioned into page nodes $\mathcal{V}^p$ and element nodes $\mathcal{V}^e$. Every node record contains a stable identifier, node type, document identifier, zero-based page index, an abstract, an embedding path, and lists of incoming and outgoing edge identifiers. Element records additionally contain an element index within the page and, when available, a bounding box. Type-specific content is stored directly in the serialized node record.

A page node stores the page size, full-page image path, and a reference to the page-level retrieval embeddings. An element node corresponds to one non-auxiliary parser block. The current builder supports the following types and fields:

`Type`	Stored type-specific content
`title`	Heading text and hierarchy level
`paragraph`	Paragraph text
`list`	List type and item texts
`image`	Local image path, extracted content, caption, and footnote
`chart`	Local image path, extracted content, caption, and footnote
`table`	Local image path, HTML, table type, nesting level, caption, and footnote
`equation_interline`	Extracted LaTeX, equation type, and optional image path
`code`	Language, code text, and caption
`algorithm`	Algorithm text and caption

Table 4: Element types represented in the offline evidence graph. Fields may be empty when the parser does not provide the corresponding content.

Page headers, page footers, page numbers, aside text, and standalone page-footnote blocks are treated as auxiliary parser output and do not become independent nodes. Captions and footnotes attached to an image, chart, or table remain part of that element. This filtering keeps repeated page furniture out of the online candidate set while retaining annotations associated with substantive visual evidence.

The two node granularities serve different roles. Page nodes preserve the complete visual layout and provide the units used for initial retrieval. Element nodes expose locally concentrated evidence and parser structure. The parent page index links every element to its visual context even when the element also has a local image or extracted text representation.

B.2 Structural Relations

The structural edge set is

$$\mathcal{E}_{\text{struct}} = \mathcal{E}_{\text{con}} \cup \mathcal{E}_{\text{read}} \cup \mathcal{E}_{\text{lay}} \cup \mathcal{E}_{\text{sec}}. \tag{6}$$

Each stored edge specifies a source, target, edge type, relation name, weight, and optional metadata.

Containment. For every element v_j^e on page v_i^p, the builder stores a directed `contains` edge $v_i^p \to v_j^e$. Its weight is the element bounding-box area when a valid box is present and 1 otherwise. At query time, elements on an active page are enumerated directly as containment candidates. Containment is therefore used to move from a retrieved page to its local blocks; the online graph store does not treat the stored edge as generally bidirectional.

Reading order. Page nodes are sorted by page index and connected to the next page. Element nodes are sorted first by page and then by their parser order, after which consecutive elements are connected by `next` edges. The latter sequence can cross a page boundary. Although each edge is stored in the forward direction, the online candidate generator may traverse reading-order edges in either direction, exposing a previous or next neighbor according to the active endpoint.

Layout adjacency. Layout edges are constructed independently within each page. Two elements are eligible only when both have bounding boxes, overlap vertically, and one lies entirely to the left or right of the other. For each source element, the builder ranks eligible targets by Euclidean distance between box centers and retains the nearest K_{lay}. The stored relation is `left_of` or `right_of`, the edge weight is $1/(1+d)$, and the raw center distance d is preserved as metadata. These horizontal relations are logically traversable in both directions during online expansion.

Section hierarchy. Within each page, title nodes are processed in parser order with a stack indexed by heading level. A heading is linked from its nearest open ancestor through `has_subsection`; subsequent non-title blocks are linked from the current heading through `contains_block`. Section edges are stored from the heading to its child, while the online graph store permits traversal toward either the parent section or the child content. Because the stack is rebuilt for each page, the current hierarchy construction does not create section edges across page boundaries.

B.3 Abstracts and Embeddings

Each node receives a compact abstract for controller previews and summary-level reader evidence. Before model-based summarization, the builder derives a short parser-based representation from the node content. It then uses the configured vision-language model to generate a grounded abstract. Visual node types—pages, images, charts, tables, and displayed equations—include an available page or local image together with extracted content; other element types use their textual fields. The summarization prompt asks the model to preserve explicit entities, values, labels, and relationships while avoiding facts not present in the source. Existing nonempty abstracts can be retained when a graph is rebuilt.

The semantic index uses token-level ColPali representations. Page nodes reuse the page embeddings produced for visual retrieval. For image, chart, table, and displayed-equation elements, the builder embeds the local image when that file is available and falls back to the element text otherwise. Other element types are embedded from their type-specific text. Long textual inputs are split into bounded chunks, and the resulting token embeddings are concatenated before semantic-neighbor scoring.

B.4 Semantic Neighbors and Edge Direction

Semantic neighbors are computed separately within the page layer and the element layer; the procedure does not create page-to-element semantic edges. For each source node u, token vectors are normalized and scored against every other node v in the same layer using the averaged MaxSim score

$$s(u,v) = \frac{1}{|Z_u|} \sum_{z \in Z_u} \max_{z' \in Z_v} z^\top z', \tag{7}$$

where Z_u and Z_v are their token embeddings. Self-similarity is excluded, and the builder retains the K_{sem} highest-scoring targets for each source. Every retained edge has relation `similar_to`, uses the MaxSim score as its weight, and records its neighbor rank.

Semantic edges are stored and followed in their selected source-to-target direction. This differs from reading-order, section-hierarchy, and horizontal layout edges, whose reverse traversal is enabled by the online graph store. A directed Top-K semantic relation need not be reciprocal. Thus, v can be a retained neighbor of u without the reverse edge being available.

The current graph-building command defaults to $K_{\text{lay}} = 1$ and $K_{\text{sem}} = 3$; both are command-line parameters and are recorded in the graph metadata. The build also records whether abstract generation and semantic embeddings were enabled, together with the source artifact paths and graph counts. The resulting `nodes.jsonl` and `edges.jsonl` files provide the immutable search space from which the query-time state in Appendix A is constructed.

C Controller and Rendering Details

This appendix specifies the query-time interfaces that connect the evidence graph to the evaluator and the LVLM reader. The controller observes a compact XML description of the current state,

converts the evaluator decision into validated state operations, and finally passes the non-pruned evidence to the renderer. The two interfaces expose different content. Opening an element provides fuller content to the evaluator; the reader XML contains the node abstract together with available location and image evidence.

C.1 CANDIDATE GENERATION

At the beginning of every controller round, candidates are regenerated from the current state. The generator first enumerates inactive or pruned elements on every active, non-pruned page as containment candidates. It then follows outgoing graph edges from active elements and follows incoming edges only for edge types marked logically bidirectional by the graph store. Reading-order, section-hierarchy, and horizontal layout relations can therefore expose neighbors in either direction; semantic relations are followed only in their stored source-to-target direction. Candidates whose target page is outside the allowed page set, whose parent page is pruned, or whose target is already active or opened are discarded. Duplicate action payloads are removed before indexing.

Each candidate contains a stable action identifier, an action type, a payload, and a short preview. Only ACTIVATENODE candidates are numbered in the evaluator XML. Their attributes identify the source node, edge type, relation, and traversal hint, while the preview is truncated to the configured character limit. The index is local to the current round and is resolved back to the stable candidate identifier before execution. This keeps the evaluator output short without requiring it to copy long graph identifiers.

The state object also contains a search-result field for compatibility with earlier graph interfaces. In the current controller path, a SEARCHEVIDENCE request is handled by the builder through a fresh page retrieval call; the retrieved pages are activated directly and become containment candidates in the next round. The builder excludes already active pages from this re-retrieval. It does not populate the search-result field as a separate candidate source in this path.

C.2 EVALUATOR INTERFACE

The evaluator receives the question, the page-organized evidence state, a bounded recent trace, and the current candidate list. For every visible page, the XML includes its abstract and all active or opened elements. Active elements expose their abstract, type, and state. Opened non-page elements additionally expose a truncated form of the graph-store text, an optional bounding box, and an image reference when a node image is available. The recent trace records successful and failed state operations, evaluator decisions, searches, pruning reasons, and watchdog events in a compact form.

The evaluator returns a structured decision with five fields:

- `stop`, a Boolean termination flag;
- `selected_actions`, containing indices of numbered ACTIVATENODE candidates;
- `open_requests`, containing active non-page node identifiers;
- `search_request`, containing at most one focused query; and
- `prune_requests`, containing visible page or element identifiers and a reason for each removal.

The parser first extracts the `agent_decision` XML and then falls back to a permissive tag parser when a response has minor formatting defects. An invalid candidate index is recorded as a validation error and does not change the state, but still counts as an attempted selection. The builder truncates numbered selections to the per-round limit and checks the cumulative stopping threshold after processing the batch. Open, search, and prune requests use dedicated fields and are processed only if this check permits the loop to continue.

C.3 STATE VALIDATION AND CONTROLLER SAFEGUARDS

The state dispatcher applies and records each operation. ACTIVATEPAGE requires an allowed page and changes that page to Active. ACTIVATENODE activates a page target directly or activates an

element after ensuring that its parent page is available. OPENNODE requires an active non-page element; page nodes cannot be opened. PRUNENODE accepts active, opened, or inactive nodes, records the supplied reason, and changes the node to Pruned. A pruned element can later be reactivated, whereas elements under a pruned page remain unavailable while their parent is pruned.

The loop terminates normally only when the evaluator sets `stop=true` and supplies no activation, opening, search, or pruning request. The trace separately records termination caused by a disabled controller, missing evaluator client, invalid evaluator output, round limit, cumulative selection threshold, or repeated no-progress rounds. The trace stores the state snapshot after each operation, the evaluator decision, candidate-index mapping, validation errors, and the final stopping reason. This record supports later trace analysis without changing the evidence graph itself.

After the online loop, the builder performs a bounded final-open pass. It ranks still-active non-page nodes by lexical overlap between the question and the node's available text fields, opens up to the configured limit, and records these operations with a distinct final-pass marker. For newly activated pages introduced by focused search, the pass may first activate and open question-relevant page elements that have not yet been processed. This pass is a rendering safeguard; it does not replace the controller's per-round activation budget.

C.4 STRUCTURED READER RENDERING

The renderer orders pages from successful page-activation trace entries, prioritizing pages introduced by the question-page scope when present and then retaining the remaining pages in activation order. Pruned pages are removed. Within each remaining page, opened and active non-page nodes are emitted once, with opened nodes listed before active nodes. A page is represented by an XML `page` block containing its abstract and, when enabled, a full-page image reference.

Each emitted node contains its type and abstract. When a bounding box is available, the renderer includes it as normalized coordinates. For a node with an image path, the renderer attaches the local image. If no local image exists but a valid bounding box and page image are available, it creates a crop from the page image. Thus, page images preserve global layout, local images preserve parser-provided visual content, and crops provide a fallback for localized evidence. Image references are tracked separately from the text XML so that the serialized trace can identify their page, node, kind, and source path.

The reader prompt instructs the LVLM to use the structured XML and interleaved document images together, to prefer visible document content when extracted text conflicts with an image, and to avoid outside knowledge. It requests one `think` block followed by one `answer` block. When the rendered evidence does not support an answer, the configured response is 'Not answerable.'. If no image is available, the renderer collapses the multimodal parts into a text-only message while preserving the same evidence structure.

The final trace records the rendered text parts, image references, content-part count, and a sanitized message representation. These records describe what was supplied to the reader for a particular query; they do not imply that every available node field or every graph edge was included in the reader context.

D EXPERIMENTAL PROTOCOL

This appendix specifies the datasets, comparison groups, reader settings, and metrics needed to interpret the reported results.

D.1 DATASETS AND EVALUATION GROUPS

LongDocURL contains 2,325 questions covering document understanding, numerical reasoning, and cross-element localization. Its reported breakdowns include evidence source, evidence-page count, and task type. MMLongBench-Doc evaluates long-PDF questions with evidence-source, single-page/cross-page, and answerability groups. The main paper reports the aggregate accuracy for LongDocURL and aggregate accuracy and F1 for MMLongBench-Doc; the supplied breakdown figure reports the additional groups.

D.2 Compared Systems

The comparison groups are Direct MLLM, Text RAG, Page-level Visual RAG, and Graph/Agentic RAG. Text RAG includes BM25 and ColBERTv2. Page-level Visual RAG includes M3DocRAG, VisRAG, and EVisRAG/VisRAG 2.0. Graph/Agentic RAG includes MLDocRAG and G^2-Reader. The reader and retrieval conditions below define the scope of these comparisons.

D.3 Reader and Retrieval Scope

The locally reproduced systems use Qwen3-VL-8B-Instruct as the reader. MLDocRAG is included using the results reported by its original paper and is marked with a dagger in Table 1; it is not part of the shared-reader claim. Retrieval is restricted to the document or benchmark-provided page scope for each question. MAGE-RAG begins with page-level retrieval, builds or loads the offline evidence graph, and performs query-time controller actions under the configured page, round, action, and final-opening limits. The reader input is grouped by page with page summaries and selected element evidence; available images or crops are included according to the renderer state.

D.4 Metrics and Result Provenance

Accuracy is the primary metric on LongDocURL. MMLongBench-Doc reports accuracy and F1, with additional single-page, cross-page, and unanswerable groups. Budget and ablation tables report percentages. Trace and logical-cost tables characterize a representative MAGE-RAG run for diagnostic inspection. In particular, a logical cost recorded for MAGE-RAG alone does not establish cost superiority over the baselines.

For the $k = 10, T = 5, a = 3$ setting, we report 50.10 accuracy, 58.61 F1, 57.97 single-page accuracy, 37.55 cross-page accuracy, and 62.29 unanswerable accuracy. The budget table and curve use this row consistently.

E Trace and Logical-Cost Analysis

The query process records an evidence-control trajectory for each question. The trajectory includes activated pages, opened nodes, focused-search actions, prune actions, controller iterations, stopping information, and the materialized reader input. These records are useful for diagnosing whether an error originates in initial retrieval, evidence expansion, controller decisions, rendering, or answer generation.

E.1 Representative Trace Statistics

Table 5 and Fig. 5 report a representative MMLongBench-Doc run. Correct and incorrect samples have similar average numbers of opened nodes, activated pages, search actions, and reader images. These aggregate counts provide little separation between the two groups, motivating inspection of the stopping reason, answer-page hits, and final reader evidence for individual errors.

Table 5: Trace statistics of MAGE-RAG on MMLongBench-Doc. Statistics are computed from a representative run with $k = 3, T = 5, a = 3$; averages are split by answer-level correct and incorrect samples.

Group	Count	Steps	Iter.	Pages	Open	Prune	Search	Images
Correct	549	64.06	3.13	5.28	17.57	1.64	1.53	21.54
Incorrect	542	61.20	3.34	5.11	16.58	1.73	1.41	20.42

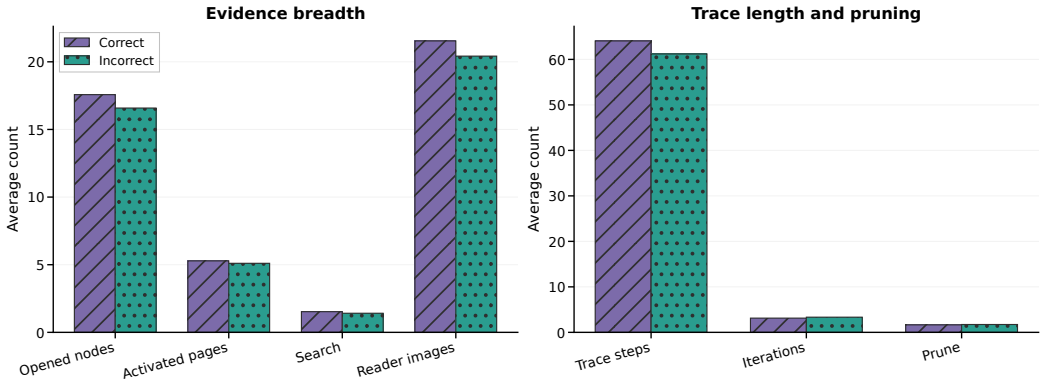

Figure 5: Trace statistics corresponding to Table 5. The panels compare average evidence breadth, trace length, controller iterations, and pruning counts for correct and incorrect samples.

E.2 Logical Input Cost

Table 6 and Fig. 6 describe logical input quantities from the same run, including images, context pages, context nodes, content parts, and LLM calls. These quantities characterize the work requested by the pipeline. Wall-clock latency also depends on concurrency, queueing, caching, and backend load, which were not controlled in this diagnostic run. A comparative efficiency claim would require measurements under matched serving conditions.

Table 6: Logical-cost statistics of MAGE-RAG on MMLongBench-Doc. Statistics are computed from a representative run with $k = 3, T = 5, a = 3$; Score/Image and Score/LLM divide the aggregate score by the corresponding aggregate cost.

Group	Count	Acc	Images	Pages	LLM	Parts	Nodes	Score/ Image	Score/ LLM
Overall	1091	52.25	20.99	4.05	3.23	49.02	21.05	0.025	0.162
Correct	549	100.00	21.54	4.11	3.13	50.19	21.59	0.046	0.320
Incorrect	542	3.87	20.42	3.99	3.34	47.83	20.50	0.002	0.012

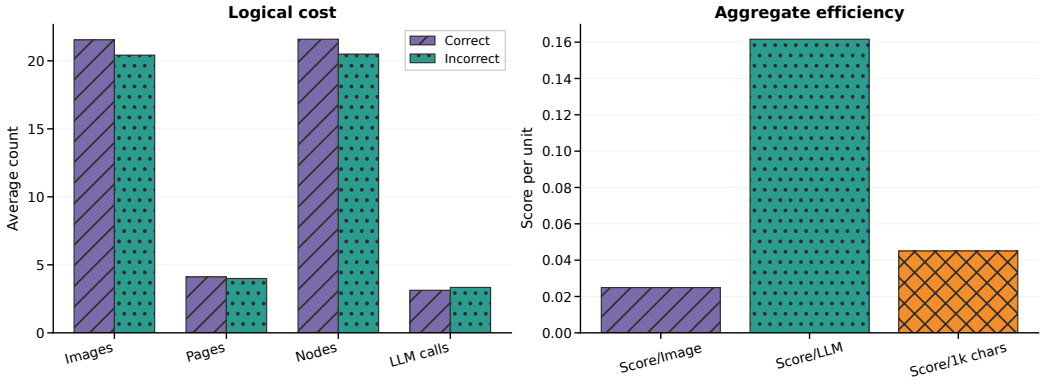

Figure 6: Logical-cost visualization corresponding to Table 6. The left panel compares average input quantities for correct and incorrect samples; the right panel reports aggregate score per unit of logical input cost.

E.3 Interpreting the Trace

The trace supports four inspection questions. First, did initial page retrieval activate a page containing answer-supporting evidence? Second, did graph expansion or focused search expose the required local or cross-page evidence? Third, did opening or pruning decisions remove evidence needed by the reader? Fourth, did the final rendered input contain enough visible and textual information for the reader to answer? These questions separate evidence-construction failures from reader failures without treating the controller trace as a proof of causal attribution.

F Representative Cases

Two recorded cases illustrate how to inspect evidence construction after a correct or incorrect answer. Their purpose is to guide individual diagnosis; estimating failure frequencies requires a larger annotated sample.

F.1 Successful Numerical Comparison

For question `q000289`, the system is asked for the average performance gap between ProgramFC and the proposed system on the HOVER dataset. Both the gold answer and prediction are 0.45. The controller reaches the watchdog limit after five iterations. Its trace contains 346 events, 100 opened nodes, five active nodes, five pruned nodes, ten searches, and 105 reader images. The recorded page sequence begins with pages 6, 5, 2, 8, 7, 3, 9, 4, 1, and 10. The answer is correct despite watchdog termination. Inspecting the materialized evidence can help determine how the requested calculation was supported before the loop ended.

F.2 Failed Figure Counting

For question `q000341`, the system must count the people in figures from pages 19–21. The gold answer is 8, whereas the prediction is 12.0. The controller again reaches the watchdog limit after five iterations. The trace contains 345 events, 100 opened nodes, six active nodes, twelve pruned nodes, ten searches, and 106 reader images. Its recorded page sequence includes pages 9, 12, 2, 3, 8, 16, 24, 7, 19, 23, 14, and 1.

The failure cannot be assigned to the controller from these aggregate fields alone. Page 19 appears in the recorded sequence, but the trace summary does not establish whether the necessary visual regions from all three target pages were preserved and rendered clearly. Diagnosing the case therefore requires checking the page-level retrieval hits, selected figure elements, prune events, final image references, and the reader response together. The trace therefore narrows the inspection path; causal attribution still requires checking the evidence and reader response.

G Reproducibility Details

The anonymous repository contains the executable pipeline described in this section. All paths below are relative to the repository root. The top-level Hydra entry point is `main.py`; benchmark configurations are stored in `configs/benchmarks/`, and baseline configurations are stored in `configs/baselines/`. The MAGE-RAG configuration is `configs/baselines/magerag.yaml`.

G.1 Preprocessing and Graph Construction

The repository README gives commands for rendering PDF pages, parsing documents, generating ColPali page and question embeddings, and building evidence graphs. The corresponding scripts are under `benchmarks/scripts/`. Each processing stage has an artifact-verification command that checks the expected document counts and files before the next stage. Evidence graphs store document metadata and graph settings in `graph.json`, node records in `nodes.jsonl`, relations in `edges.jsonl`, and element embeddings in Safetensors files.

G.2 Evaluation Entry Point

After the model services and benchmark artifacts are available, both evaluations can be launched with:

```
uv run python main.py --multirun \
  benchmarks=mmlongbench,longdocurl \
  benchmarks.workers=16
```

The worker count may be adjusted to the available serving capacity. Per-question predictions are written under `results/<benchmark>/magerag/`; aggregate metrics use matching `*.metrics.json` files, and Hydra stores resolved configurations and logs under `outputs/`. The runner supports resuming interrupted evaluations.

G.3 External Resources

The pipeline requires the benchmark PDFs and question files, a document parser, ColPali page embeddings, and an OpenAI-compatible multimodal model service. The reference setup uses Qwen3-VL-8B-Instruct for reading and control and `vidore/colpali-v1.3-hf` for visual retrieval. Dataset files, model weights, and service credentials remain subject to their original providers' licenses and access procedures; they are not embedded in the paper source package.