DocAtlas: Long-Document Understanding as Mutable-State Interaction

Hongchen Wei[1,*] **Yuanzhe Wang**[3,*] **Bei Liu**[2,†,‡] **Yifan Yang**[2]
Qi Dai[2] **Kai Qiu**[2] **Yunsheng Li**[3] **Dongdong Chen**[3]
Chong Luo[2] **Zhenzhong Chen**[1,‡] **Baining Guo**[2]

[1]Wuhan University [2]Microsoft Research Asia [3]Microsoft

Abstract

Long-document understanding requires models to find and combine evidence across many pages, layouts, tables, figures, and charts. Existing retrieval-augmented systems usually select evidence from a static index before generation, while recent agentic systems add multi-turn tool use but often rely on frozen proprietary backbones whose behavior is set by prompts. We present DocAtlas, a system that treats long-document understanding as a mutable-state information-seeking process. We instantiate DocAtlas as a mutable document harness: an external environment that determines what document information is searched, read, stored, reviewed, and shown to the model at each step. Given a document and question, the harness exposes search, reading, note-taking, and review tools, maintains a hierarchical tree and note store, and updates both as the agent records evidence. DocAtlas combines self-improving retrieval, selective evidence access, and active working memory under a fixed context budget. The same harness supports inference-time use with large vision-language models (VLMs) and end-to-end reinforcement learning (RL) for compact VLM agents. With GPT-5.4, DocAtlas reaches 71.4% on MMLongBench-Doc, exceeding the human-expert reference of 65.8%. A Qwen3.5-4B VLM trained with end-to-end RL in the DocAtlas environment reaches 63.7%, compared with a 54.4% direct-input baseline, showing that mutable document-harness design can improve compact document agents by a large margin.

Project page: `https://officeintelligence.github.io/docatlas/`.

1 Introduction

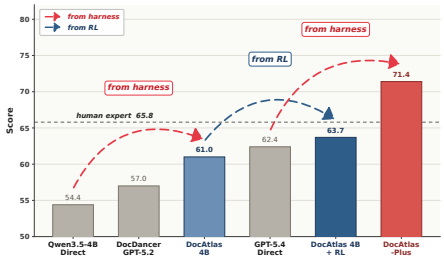
Real-world documents such as financial reports, legal contracts, scientific papers, and government filings often spread important information across dozens or even hundreds of pages. They also combine free text with tables, figures, and charts in varied layouts. Answering natural-language questions over these documents therefore requires finding, extracting, and combining evidence that is distributed across pages and modalities [6]. This remains difficult because relevant evidence may appear anywhere in a long document, visual elements such as charts and tables cannot be handled by text extraction alone [3], and complex questions often require evidence from multiple distant regions or modalities [21]. Simply feeding the full document into a model quickly runs into the practical limits of current vision-language models [12, 18, 36]. Even when the input fits within the context window, performance often drops because irrelevant content competes for attention [17]. These issues call for systems that can navigate long documents selectively, ground answers in both text and visual content, and combine information across distant parts of a document.

* Equal contribution and work done during internship at MSRA. † Project leader. ‡ Corresponding authors: Bei Liu and Zhenzhong Chen.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Method	Mutable state	Flexible interaction	Grounded memory	Trainable RL policy
M3DocRAG[3]	×	×	×	×
DocAgent[31]	×	△	×	×
SimpleDoc[13]	×	△	△	×
DocLens[43]	×	✓	✓	×
DocDancer[40]	△	✓	×	△
MACT[39]	△	△	×	×
DocAtlas	✓	✓	✓	✓

(a) Design comparison. ✓: yes, △: partial, ×: no.



(b) MMLongBench-Doc ALL.

Figure 1: Motivation and positioning of DocAtlas. (a) ✓, △, and ×denote full, partial, and absent support, respectively. Flexible interaction means that the agent can choose the order and arguments of tool calls rather than following a fixed pipeline; grounded memory means that source-attributed evidence can be stored and queried in later steps. DocAtlas combines mutable state, flexible interaction, grounded memory, and a trainable compact policy. (b) On MMLongBench-Doc, DocAtlas-Plus (our GPT-5.4 instantiation) exceeds the human-expert reference, while the same environment improves Qwen3.5-4B [29] from direct input to DocAtlas 4B and then to DocAtlas 4B+RL.

Early work [4, 8] has addressed part of this problem with retrieval-augmented generation (RAG). In these systems, an embedding-based retriever selects a fixed set of relevant pages, and a vision-language model (VLM) generates an answer in one pass [3, 11]. This design works reasonably well for simple lookups, but it gives the model no control over which pages to inspect more carefully or how to revise the search based on partial findings. To overcome this limitation, recent work has moved toward agentic methods, where a vision-language model interacts with a document through iterative tool use for searching, reading, and reasoning [13, 31, 33, 35, 39, 43]. However, these systems still depend on frozen proprietary backbones, and their agent behavior is largely specified through prompting rather than learned from data. A natural next question is whether the agent itself can be trained. DocDancer [40] takes a simpler approach by training a single open-source model on synthesized trajectories with supervised fine-tuning. However, DocDancer uses a text-only LLM as the controller, and its `Read` tool relies on an external VLM for visual understanding. As a result, the agent learns when to call a tool, but not how to interpret visual content itself. Because it is trained by imitation, it is also limited by the coverage of the expert trajectories it observes.

DocAtlas addresses these gaps by exposing the document interface as a mutable environment in which the agent policy can be trained. Equivalently, DocAtlas can be viewed as a mutable document harness: it wraps a VLM with tools and state that determine what document information is searched, read, stored, reviewed, and shown at each step. The agent does more than call tools over a fixed document representation. Each episode keeps an environment state with the document, a hierarchical tree, a structured note store, and the set of explored pages. SEARCH reads this updated state; READ exposes selected pages as Markdown, crops, and page images; NOTE records source-attributed findings and writes them back to the tree; and REVIEW revisits prior notes when later reasoning requires them. This loop makes later retrieval depend on earlier evidence and helps the agent reason across many steps under a fixed context budget. Because the same interaction protocol is used at inference time and during RL, DocAtlas also provides a direct way to train compact VLM agents instead of treating tool use as a fixed prompt script. With GPT-5.4 [26], DocAtlas reaches 71.4% on MMLongBench-Doc [21], surpassing the human-expert reference at 65.8%. A Qwen3.5-4B [29] VLM fine-tuned with end-to-end RL in the same environment reaches 63.7%, compared with a 54.4% direct-input baseline.

This work makes three contributions:

- We formulate long-document understanding as a mutable-state information-seeking process in which reading and note-taking update the retrieval and memory state that informs subsequent decisions.
- We instantiate this formulation in DocAtlas, a harness that combines self-improving retrieval, decoupled search and reading, and structured note and review operations.
- We show that the same environment supports both large VLM agents at inference time and end-to-end RL for compact VLMs: DocAtlas reaches 71.4% on MMLongBench-Doc with GPT-5.4, while an RL-tuned Qwen3.5-4B reaches 63.7% against a 54.4% direct-input baseline.

2 Related Work

Multimodal Retrieval for Long-Document Understanding. Multi-page document QA [10, 24, 34, 42] is commonly handled with multimodal RAG: pages are embedded as visual or textual vectors, a fixed top-k subset is retrieved, and a VLM answers in one pass. Recent visual retrievers and benchmarks, including ColPali [7], VisRAG [38], ViDoRe [22], and MIRACL-VISION [27], show the value of page-image retrieval for documents with complex layouts and visual content. M3DocRAG [3] and MDocAgent [11] build long-document QA pipelines on this static-index design with visual or parallel text–image retrieval. Adaptive retrieval has also been studied in text QA through query rewriting [20] and self-reflective retrieval control [1]. These methods adapt queries or retrieval decisions, but the document index itself remains precomputed. DocAtlas instead updates the retrieval state within an episode by writing page-grounded findings back into a hierarchical tree.

Agentic Document Understanding. Recent systems [2, 9, 15, 16] move from one-shot retrieval to tool-mediated interaction: Doc-React iterates sub-queries [33], SimpleDoc combines visual embeddings with page summaries [13], and DocLens separates page navigation from element localization [43]. Closest to our tree-based navigation, DocAgent [31] builds a structured XML outline with section hierarchy, page ranges, paragraph hints, captions, and identifiers for fetching full content. Its outline is primarily a fixed navigation scaffold, whereas DocAtlas treats the tree as per-question mutable state enriched by NOTE and observed by later SEARCH calls. Existing document agents can be viewed as hand-designed harnesses that decide what information is retrieved, shown, and carried forward, consistent with recent LLM harness work [14, 19]. However, they typically rely on frozen proprietary backbones, follow staged tool flows, or leave the document state unchanged. Learning-based systems such as DocDancer [40] train tool-use behavior, but either spread credit across multiple agents or rely on an external VLM for visual reading. DocAtlas puts visual reading, control, mutable memory, and RL optimization in a single VLM policy.

3 Method

Passing an entire long document to a VLM quickly exceeds the effective context window. Even when the document fits, irrelevant content can dilute attention and hurt performance [17]. Static retrieval selects a fixed subset of pages, but it cannot refine the search as evidence accumulates. We instantiate DocAtlas as a mutable document harness. Given a document and question, the harness exposes a set of tools, maintains a document tree and note store, and updates both as the agent searches, reads, and records evidence. Unlike a static retrieval harness, the state seen by later tool calls depends on earlier interactions. We describe this operationally as a mutable-state information-seeking process: the agent's actions reveal document content and update the retrieval and memory state used by later steps. This interface couples retrieval, evidence memory, and policy optimization within one environment for inference and training.

Terminology. We use *model* or *backbone* for the underlying VLM weights, *policy* for the mapping from interaction history to the next action, and *agent* for the policy running inside the DocAtlas tool environment. Thus, the same VLM can be described as a model when discussing its architecture or size, as a policy when discussing RL or action choices, and as an agent when it interacts with tools and environment state.

Given a document $\mathcal{D} = \{d_1, \ldots, d_N\}$ consisting of N pages and a natural-language query q, the agent produces a trajectory

$$\tau = (q,\ u_1, o_1,\ u_2, o_2,\ \ldots,\ u_T, a), \tag{1}$$

where each action u_t for $t < T$ is a tool call $(\texttt{tool}_t, \texttt{arg}_t) \in \mathcal{T} \times \mathcal{A}$, and the trajectory ends with a terminal action $u_T = \texttt{finish}(a)$ that outputs the final answer a. For each tool call, the environment returns an observation $o_t = \texttt{env}(\texttt{tool}_t, \texttt{arg}_t; \mathcal{S}^{(t)})$, where $\mathcal{S}^{(t)} = (\mathcal{D}, \mathcal{G}^{(t)}, \mathcal{M}^{(t)}, \mathcal{H}^{(t)})$ is the environment state comprising the document $\mathcal{D}$, the document index $\mathcal{G}^{(t)}$ (defined in §3.1), the note store $\mathcal{M}^{(t)}$ (§3.3), and the set of previously explored pages $\mathcal{H}^{(t)} \subseteq [1, N]$. The tool set $\mathcal{T} = \{\text{SEARCH}, \text{READ}, \text{NOTE}, \text{REVIEW}\}$ has a fixed action space but no fixed execution order: unlike staged pipelines, the agent may invoke any tool at any step and interleave search, reading, and memory operations as needed.

The key departure from static retrieval is that $\mathcal{S}^{(t)}$ is mutable. In particular, $\mathcal{G}^{(t)}$ and $\mathcal{M}^{(t)}$ are updated by the agent's own actions: NOTE writes evidence-grounded annotations into the tree and archives

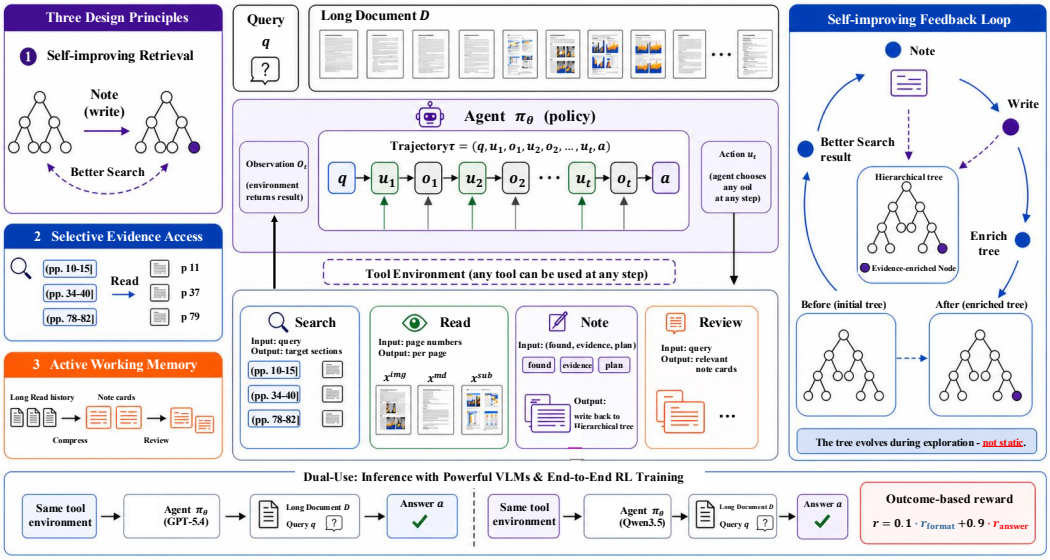


Figure 2: Overview of DocAtlas. The agent searches a mutable tree, reads selected document pages in multimodal form, writes structured notes that both compress context and update the retrieval state, reviews prior findings when needed, and finally produces an answer. Unlike static RAG, evidence gathered during reading changes the state used by later search and reasoning steps.

structured notes, while REVIEW exposes selected notes back to the policy. Thus, a later SEARCH call depends on the original document, the query, and what the agent has already read and recorded. This makes retrieval part of the closed-loop decision process rather than a fixed preprocessing step.

DocAtlas is built around a tool environment with three design principles (Figure 2). Self-improving retrieval (§3.1) organizes the document as a hierarchical tree that SEARCH navigates and NOTE updates during exploration. Selective evidence access (§3.2) separates finding evidence from consuming it: SEARCH identifies relevant sections, and READ lets the agent decide which pages to inspect and in which modality. Active working memory (§3.3) is implemented through NOTE and REVIEW, which record, optionally archive, and review evidence so the agent can reason across many steps under a fixed context budget. This interaction can be viewed as a sequential decision problem, which we use in §3.4 for end-to-end reinforcement learning.

3.1 Self-Improving Retrieval

Standard embedding-based retrieval maps each page to a fixed-dimensional vector. This discards the hierarchical structure of long documents and yields a static index that cannot improve as the agent gathers evidence. Inspired by PageIndex[1], which builds a hierarchical tree index and performs LLM-guided tree search, DocAtlas uses a tree-structured index for semantic navigation. We extend this idea in two ways: tree construction incorporates visual parsing for multimodal document understanding, and SEARCH is decoupled from READ so retrieval proposes candidate regions rather than directly determining the consumed evidence.

Hierarchical index. Each document $\mathcal{D}$ is organized offline into a tree $\mathcal{G} = (V, E)$, where each node $v \in V$ stores $(title_v,\ [p_s^v, p_e^v],\ summary_v,\ findings_v)$, namely a section title, page range, summary, and a mutable findings list (initially empty). The tree is constructed once per document by a VLM in a question-agnostic manner: the model receives structured Markdown extracted from the document together with parsed visual content, including figure/table/chart captions and layout-derived cues, and outputs a JSON tree with section titles, page ranges, and summaries. This preprocessing does not use benchmark questions, answers, or evidence annotations.

[1] https://github.com/VectifyAI/PageIndex

Structure-aware navigation. SEARCH takes a search query q_s (which may differ from q, e.g., a sub-question) and the current index state:

$$\textsc{Search}(q_s;\, \mathcal{G}^{(t)}, \mathcal{H}^{(t)}) \;\rightarrow\; \{v_1,\, \ldots,\, v_k\}. \tag{2}$$

An auxiliary LLM receives the serialized tree, including titles, page ranges, summaries, and accumulated annotations, together with q_s and explored pages $\mathcal{H}^{(t)}$. It then selects tree nodes by using the hierarchy, preferring unvisited regions when useful, and using annotations from prior exploration.

Evidence-grounded enrichment. When the agent records findings via NOTE, the evidence entries $\texttt{evidence}_t \subseteq \{(e_j, p_j, \text{type}_j)\}$ (Eq. 5) are propagated back to the tree. For each cited page p, the finest-grained covering node $v^* = \arg\min_{v:\, p \in [p_s^v, p_e^v]} |p_e^v - p_s^v|$ receives a compact annotation in $findings_{v^*}$. The index thus evolves:

$$\mathcal{G}^{(t+1)} = \texttt{enrich}\big(\mathcal{G}^{(t)},\; \texttt{evidence}_t\big), \tag{3}$$

so later SEARCH calls see accumulated findings alongside original summaries. Each annotation contains a normalized finding, its source page, evidence type, and the step at which it was observed. The enrichment operation is deliberately conservative: it appends or merges source-attributed findings into the finest covering node but never rewrites the original section summary. Annotations are navigation hints only and do not replace direct evidence when composing the final answer.

3.2 Selective Evidence Access

In most existing agentic systems, retrieval and reading are tightly coupled: if search returns a candidate set $C = \{v_1, \ldots, v_k\}$, all pages in $\texttt{pages}(C) \triangleq \bigcup_{v \in C} [p_s^v, p_e^v]$ are consumed. This wastes context when a relevant section spans many pages and prevents accessing pages outside C when evidence is referenced indirectly. DocAtlas decouples these stages: SEARCH proposes candidate regions, while READ gives the agent explicit control over the final evidence set $P \subseteq [1, N]$. Retrieval results are therefore proposals rather than mandatory context; the agent may read pages from a search hit, neighboring pages, explicit page references, or unresolved gaps recorded in memory.

Multimodal reading. The agent selects a page subset P and invokes:

$$\textsc{Read}(P) \;\rightarrow\; \big[(x_p^{\text{img}},\; x_p^{\text{md}},\; x_p^{\text{sub}})\big]_{p \in P}, \tag{4}$$

returning, for each page p, three content types: a full-page layout image x_p^{img}, a structured Markdown rendering x_p^{md} (extracted by MinerU [25]), and a set of cropped sub-images $x_p^{\text{sub}} = \{x_{p,1}^{\text{sub}}, \ldots\}$ for figures and charts. The read observation is assembled adaptively rather than fixed by backbone type. By default, the agent receives the page layout image together with Markdown, so text, layout, and global spatial context are jointly available. For questions that require fine-grained visual inspection, READ further attaches cropped sub-images to zoom in on the relevant regions. This makes READ the main channel through which visual observations enter the agent and the tool whose use can be improved by outcome-based RL.

3.3 Active Working Memory

Each READ invocation appends multimodal observations to the interaction history h_t. At step t, the cumulative token cost $\sum_{i \le t}(|u_i| + |o_i|)$ grows toward the context budget B, forcing a tradeoff between retaining early evidence and allocating tokens to further exploration. DocAtlas therefore separates active context from evidence memory: bulky page observations remain in the interaction history when available, while NOTE stores compact, source-attributed findings that can be retrieved later by REVIEW.

Structured note-taking. NOTE records findings as a structured tuple

$$n = \big(\texttt{found},\;\; \texttt{evidence},\;\; \texttt{plan}\big), \tag{5}$$

where `found` summarizes what has been established, $\texttt{evidence} = \{(e_j, p_j, \text{type}_j)\}_{j=1}^{J}$ is an array of typed, source-attributed entries ($\text{type}_j \in \{\text{text}, \text{table}, \text{image}\}$), and `plan` states remaining information gaps. Unlike generic transcript summarization, notes in DocAtlas are structured evidence objects: they store what has been established, where it was observed, what modality supports it, and what

remains unresolved. NOTE triggers two side effects. When archival is enabled, prior READ outputs in h_t can be replaced in-place with short placeholders, yielding $\hat{h}_t$ with $|\hat{h}_t| \ll |h_t|$. Evidence entries are also written back to the tree (Eq. 3), closing the feedback loop with SEARCH.

To avoid turning memory into an unconstrained summary channel, notes are constrained to be extractive and page-grounded: each evidence entry must be copied or normalized from a previously read page and paired with its source page and modality. Tree annotations therefore act as source-attributed navigation hints rather than free-form model beliefs. We use append-and-merge semantics: new entries are concatenated with existing ones rather than replacing them, so the auxiliary model can inspect the accumulated evidence trail during later search.

Evidence review. As notes accumulate in $\mathcal{M}^{(t)}$, $\textsc{Review}(q_v)$ reviews the note store and returns only the findings relevant to a query q_v. Each note $n_i \in \mathcal{M}^{(t)}$ is projected to a compact note card $c_i = (\texttt{id}_i, \texttt{step}_i, \texttt{found}_i, \texttt{pages}_i)$ that strips the full evidence payload. An auxiliary LLM receives $\{c_1, \ldots, c_{|\mathcal{M}^{(t)}|}\}$ and selects the relevant subset (greedy decoding; prompt in Appendix A.7). REVIEW exposes source-attributed evidence summaries to the policy without requiring the full multimodal transcript to remain in context.

3.4 End-to-End RL over Agent Trajectories

Because DocAtlas exposes document understanding as a sequence of structured actions and observations, it can train compact VLM agents directly. We optimize a single VLM policy over complete tool trajectories using outcome rewards. The tree-building procedure, tool executors, and auxiliary models used by SEARCH and REVIEW remain fixed during policy optimization; the policy learns when to search, what to read, what to record, and when to answer.

Formulation. The agent policy π_θ is a single VLM that, at each step t, conditions on $h_t = (q, u_1, o_1, \ldots, u_{t-1}, o_{t-1})$ and emits $u_t \sim \pi_\theta(\cdot \mid h_t)$. During RL training, each action is either a structured tool call or a terminal answer normalized as `\boxed{}` for reward parsing; the trajectory terminates at the answer or after $T_{\max}$=8 turns. The tool implementations and auxiliary models receive no gradient updates; runtime tree and memory states remain mutable.

Reward and optimization. RL rewards are computed from the final answer only. If the last visible answer lacks a parsable `\boxed{}` field, the reward is zero; otherwise, we extract the boxed content and apply the LongDocURL type-aware answer score:

$$r = \mathrm{Score}_{\mathrm{LongDocURL}}\big(\mathrm{ExtractBoxed}(a), a^\star\big). \tag{6}$$

We optimize with GRPO [30] using DAPO-style asymmetric clipping [37] (ϵ_{low}=0.1, ϵ_{high}=0.3), n=8 rollouts, group-normalized advantages, and token-level KL regularization (λ_{KL}=0.01). Appendix A.5 gives full reward and training details.

4 Experiments

We organize the evaluation around two questions: **Q1:** Does a mutable tree-based tool environment improve long-document understanding on benchmarks with complex layouts and visual content? **Q2:** Can the same environment train compact VLM policies through end-to-end RL, instead of only serving as an inference-time prompting scaffold? We report results for large VLM agents at inference time and RL-trained Qwen3.5 policies.

4.1 Experimental Setup

Benchmarks and metrics. We evaluate on three widely used long-document understanding benchmarks and follow their standard evaluation protocols. MMLongBench-Doc [21] reports overall accuracy, F1, LLM-as-judge score, and accuracy by evidence source. FinRAGBench-V [41] reports LLM-as-judge scores overall and by evidence type. LongDocURL [5] reports the overall LLM-as-judge score; we use GPT-5.4 as the judge. Since LongDocURL is used to construct the RL training set (§4.1), we do not report Qwen RL results on LongDocURL.

DocAtlas configuration. The environment exposes four tools, SEARCH, READ, NOTE, and REVIEW, with no fixed execution order. READ returns structured Markdown, layout images, and cropped

Table 1: Main results on MMLongBench-Doc [21], FinRAGBench-V [41], and LongDocURL [5]. MMLongBench-Doc is reported by evidence source: text (TXT), layout (LAY), chart (CHA), table (TAB), figure (FIG), unanswerable (UNA), with overall accuracy (Acc), F1, and LLM-as-judge score (LasJ). FinRAGBench-V reports LLM-as-judge scores by evidence type and overall. LongDocURL reports overall LLM-as-judge score using GPT-5.4. Compact DocAtlas rows use GPT-5.4 as a fixed reference auxiliary for SEARCH/REVIEW; Table 4 tests open-weight and self auxiliaries. DocLens and † rows are copied from prior papers and are included for comparison; unavailable entries are marked with –. Qwen RL rows omit LongDocURL because LongDocURL is used for RL data construction. Green cells mark the best-performing method in each metric column, and pale yellow cells mark the second-best-performing method; * denotes overall accuracy above the MMLongBench-Doc human-expert reference.

Model	MMLongBench-Doc									FinRAGBench-V				LongDocURL
	TXT	**LAY**	**CHA**	**TAB**	**FIG**	**UNA**	**Acc**	**F1**	**LasJ**	**TXT**	**TAB**	**CHA**	**LasJ**	**LasJ**
Vanilla VLMs														
GPT-5.2	45.8	45.6	45.3	43.3	33.5	**86.0**	52.8	52.1	57.6	40.4	28.9	40.7	36.0	59.4
GPT-5.4	57.8	57.7	55.7	61.1	51.7	75.6	62.4	59.4	63.4	64.6	45.2	59.9	55.1	66.9
Claude-4-Sonnet	50.4	49.4	50.5	57.3	43.9	59.0	53.4	–	–	36.6	20.2	51.9	33.8	–
Gemini-2.5-Flash	44.0	53.2	46.0	43.9	48.2	56.7	49.6	–	–	49.0	41.6	41.0	43.0	–
Gemini-2.5-Pro	52.1	62.1	55.5	55.3	54.0	59.9	58.1	–	–	62.2	55.3	50.4	54.9	–
Qwen3.5-4B	47.5	48.7	47.6	55.4	45.5	63.4	54.4	53.1	58.7	66.2	49.0	48.8	52.8	52.4
Qwen3.5-9B	54.2	53.8	51.0	55.7	46.4	66.9	58.0	55.0	60.2	64.2	52.2	54.7	55.8	55.5
Qwen3.5-397B-A13B	–	–	–	–	–	–	61.9	–	–	–	–	–	–	–
VLMs Augmented with OCR														
Claude-4-Sonnet	52.7	51.6	50.0	58.1	45.3	65.9	56.0	–	–	58.7	21.6	54.3	41.0	–
Gemini-2.5-Flash	55.9	54.9	52.7	63.4	50.3	60.8	58.5	–	–	67.6	64.4	46.1	58.3	–
Gemini-2.5-Pro	59.7	65.3	60.8	68.3	55.7	58.4	63.3	–	–	70.0	70.0	56.2	64.9	–
VLM-based Agentic Frameworks														
M3DocRAG (w/ Qwen2-VL-7B)†	30.0	23.5	18.9	20.1	20.8	5.8	21.0	–	–	–	–	–	–	–
MDocAgent (w/ GPT-4o)†	–	–	–	–	–	–	42.0	–	–	–	–	–	–	–
DocDancer (w/ GPT-5.2)	–	–	–	–	–	–	57.0	–	–	–	–	–	–	–
SimpleDoc														
w/ Claude-4-Sonnet	52.1	53.3	58.3	62.4	46.9	66.5	58.6	–	–	59.6	68.9	54.9	61.7	–
w/ Gemini-2.5-Flash	45.5	57.4	49.0	51.6	45.2	66.5	53.3	–	–	70.2	56.2	53.6	58.3	–
w/ Gemini-2.5-Pro	48.4	54.8	55.7	56.1	52.5	59.7	56.6	–	–	67.5	64.0	60.9	63.6	–
DocLens														
w/ Claude-4-Sonnet	59.9	58.2	54.4	63.9	55.3	74.0	63.3	–	–	70.2	66.0	60.3	64.8	–
w/ Gemini-2.5-Flash	59.5	61.5	54.8	66.9	59.0	73.8	64.7	–	–	69.9	71.3	64.5	68.5	–
w/ Gemini-2.5-Pro	63.7	64.6	64.3	69.7	60.2	72.2	67.6*	–	–	68.9	74.2	67.1	70.4	–
DocAtlas (Ours)														
w/ Qwen3.5-4B	58.4	54.6	56.1	67.0	48.1	70.6	61.0	58.7	63.5	73.3	66.5	65.4	67.9	72.5
w/ Qwen3.5-9B	58.8	55.3	56.2	67.8	51.9	70.8	61.6	59.4	64.7	73.9	67.1	66.4	69.8	74.0
RL w/ Qwen3.5-4B	59.2	55.9	57.5	67.8	51.8	74.6	63.7	62.4	67.9	75.2	70.7	68.5	71.7	–
RL w/ Qwen3.5-9B	66.6	60.5	58.9	70.1	59.9	59.3	64.4	63.1	69.6	75.8	72.0	70.7	72.6	–
w/ GPT-5.2	67.0	**71.5**	68.0	**74.4**	67.6	66.5	70.6*	69.6	73.9	76.0	76.2	73.1	75.2	77.5
w/ GPT-5.4	**68.8**	69.0	**68.2**	74.1	**69.8**	68.6	**71.4***	**70.3**	**74.6**	**76.3**	**77.4**	**73.2**	**75.6**	**78.8**
Human Expert†	–	–	–	–	–	–	65.8	–	–	–	–	–	–	–

sub-images. NOTE records extractive, page-grounded evidence and optionally archives bulky read observations from the active context. The auxiliary LLMs inside SEARCH and REVIEW are frozen routing and memory-selection modules that can be swapped without changing the policy or tools; the main table uses GPT-5.4 as a fixed reference, and §4.4 tests open-weight and self auxiliaries. Archival is enabled for RL-trained compact policies because unbounded accumulation of tool observations makes multi-rollout training unstable (§4.1).

RL data construction and training. We construct RL data from LongDocURL using model-specific pass@16 filtering. For each compact VLM policy, we sample 16 trajectories per question in the DocAtlas environment and retain medium-difficulty questions with at least one but not all correct rollouts; all-correct questions provide little advantage signal, while all-wrong questions provide no positive trajectory under sparse rewards. Because filtering is performed separately for each policy, the retained sets differ: 612 questions for Qwen3.5-4B and 587 for Qwen3.5-9B. In early RL runs,

Table 2: Same-backbone comparisons on MMLongBench-Doc. Each pair uses the same backbone. DocLens scores are from the published evaluation [43].

Harness / backbone	TXT	LAY	CHA	TAB	FIG	UNA	Acc	F1	LasJ
DocLens / Gemini-2.5-Pro	63.7	64.6	64.3	69.7	60.2	72.2	67.6	–	–
DocAtlas / Gemini-2.5-Pro	68.8	70.5	66.4	71.3	70.9	66.0	69.8	69.4	72.9
Codex / GPT-5.4	69.0	69.2	66.5	74.3	69.6	60.0	69.5	69.1	72.6
DocAtlas / GPT-5.4	68.8	69.0	68.2	74.1	69.8	68.6	71.4	70.3	74.6

the multimodal output of READ caused highly variable and rapidly growing contexts across rollouts, making memory use unstable. We therefore enable context archival during RL training: after the agent records a page-grounded note, bulky read observations can be replaced by short placeholders while the structured evidence remains accessible through REVIEW. This reduces model input length and makes training more stable. Appendix A.4 gives the episode-level algorithm flow, and Appendix A.5 gives the reward parsing and RL hyperparameters.

4.2 Main Results

Table 1 summarizes the results on all three benchmarks. Long-document performance is not determined by the backbone alone: GPT-5.4 rises from 62.4 direct accuracy on MMLongBench-Doc to 71.4 inside DocAtlas, a 9.0-point gain over direct input and 5.6 points above the reported human-expert reference of 65.8. The gain appears across answerable evidence types, especially visual and structured cases, while unanswerable accuracy decreases from 75.6 to 68.6. Unanswerable performance varies across policies: the 4B policy improves from 70.6 to 74.6 after RL. The same pattern appears beyond MMLongBench-Doc: GPT-5.4 improves from 55.1 to 75.6 LasJ on FinRAGBench-V and from 66.9 to 78.8 on LongDocURL; GPT-5.2 also reaches 70.6 on MMLongBench-Doc and 77.5 on LongDocURL.

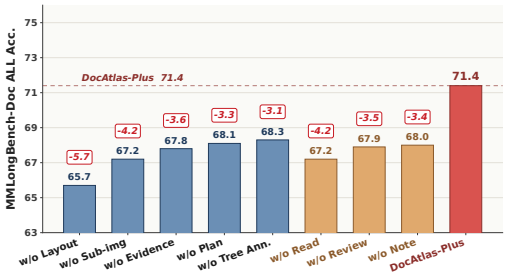
In the fixed-reference auxiliary setting, the mutable environment also improves compact open policies. Qwen3.5-4B improves from 54.4 with direct input to 61.0 with DocAtlas, and Qwen3.5-9B improves from 58.0 to 61.6. The gains are strongest on tables and other structured evidence, where the agent can first localize candidate regions and then inspect a smaller set of pages or crops. This suggests that selective multimodal access helps compact models more than simply exposing the whole document in one context.

RL further improves compact policies beyond inference-time tool prompting. On MMLongBench-Doc, Qwen3.5-4B moves from 61.0 to 63.7 after RL, and Qwen3.5-9B from 61.6 to 64.4. The gains transfer beyond the RL data source: FinRAGBench-V improves from 67.9 to 71.7 for 4B and from 69.8 to 72.6 for 9B, while Qwen RL rows omit LongDocURL because it is used for data construction. The category changes indicate that RL is not only changing the final answer style; it improves the policy's choices about when to search, read, write notes, and review them. The GPT-5.4 auxiliary is not required by the method: §4.4 shows that open-weight or self auxiliaries keep most of the compact-policy gain.

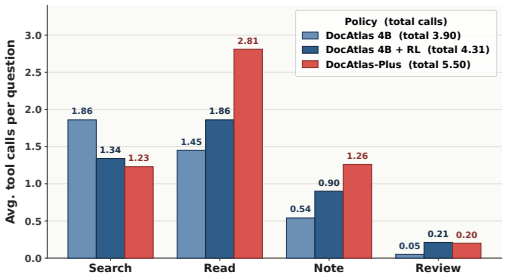
Same-backbone comparisons. Table 2 compares DocAtlas with DocLens on Gemini-2.5-Pro and with Codex on GPT-5.4. With Gemini-2.5-Pro, DocAtlas reaches 69.8 accuracy versus the published DocLens result of 67.6 [43]. With GPT-5.4, DocAtlas reaches 71.4 versus 69.5 for Codex evaluated with the same document interface and reasoning effort. The gains are 2.2 and 1.9 points, respectively. The GPT-5.4 comparison also improves F1 from 69.1 to 70.3 and LasJ from 72.6 to 74.6.

4.3 Ablation Study

Figure 3(a) ablates DocAtlas on MMLongBench-Doc; Appendix A.2 gives the full category-level table. The largest losses come from visual access. Removing layout images lowers ALL from 71.4 to 65.7, with large drops on figures and charts; removing cropped sub-images lowers ALL to 67.2. Full-page layout images preserve spatial context and cross-element relations, while cropped sub-images expose local details inside figures, charts, and tables. The result supports the two-level



(a) Component ablation.



(b) Average tool calls per question.

Figure 3: Component and tool-use analysis on MMLongBench-Doc. (a) Bars report ALL accuracy after removing either an entire tool or an internal component of a tool; full DocAtlas with GPT-5.4 reaches 71.4. Detailed category-level results are in Appendix A.2. (b) After RL, the 4B policy uses fewer SEARCH calls and more READ, NOTE, and REVIEW calls despite only a small increase in total calls; DocAtlas-Plus denotes the GPT-5.4 instantiation.

design of READ: neither text-only extraction nor a single visual scale is sufficient for long documents with complex layouts and visual content.

Selective evidence access also matters. Without READ, the agent consumes SEARCH candidates directly and ALL falls to 67.2, showing that SEARCH should remain a high-recall proposal mechanism rather than the final evidence selector. The mutable-state components each contribute consistent gains: removing the evidence field, REVIEW, NOTE, plan field, or tree annotation yields 67.8–68.3 ALL. These similar drops show that memory is a pathway rather than one switch: notes preserve page-grounded facts, review retrieves them across steps, and tree annotations feed local findings back into later search.

4.4 Behavior and Robustness Analysis

This section checks whether the gains in Table 1 come from the intended behavior: learned tool use, efficient evidence access, and robustness to the auxiliary model.

Tool-use behavior. Figure 3(b) shows that RL changes the compact policy's tool-use pattern. Total calls grow by only 10.5%, but the mix shifts sharply: SEARCH decreases by 28.0%, while READ, NOTE, and REVIEW increase by 28.3%, 66.7%, and 320.0%. The policy is not just trying more actions; it shifts effort from broad localization to reading pages, recording evidence, and checking saved notes. The clearest change is REVIEW: the no-RL 4B policy almost never uses it (0.05 calls/question), whereas the RL policy reaches 0.21, close to GPT-5.4's 0.20 rate. This suggests that RL teaches the compact model to use the harness's note-and-review pathway rather than treating each read as an isolated context extension.

Table 3: Search–Read evidence efficiency on MMLongBench-Doc. Avg. is the mean number of pages; All-Hit is the fraction of examples covering all gold evidence pages; F1 is page-level evidence F1.

Method	Avg.	All-Hit	F1
ColQwen top-2	2.00	64.12	38.75
ColQwen top-6	6.00	76.42	24.36
ColQwen top-10	10.00	83.60	18.38
Full SEARCH	16.65	**87.45**	38.33
w/o ANNOTATION	16.94	80.05	29.40
First READ	2.67	59.90	56.04
w/o ANNOTATION	2.61	58.21	56.37
Full READ	5.79	78.01	**58.99**
w/o ANNOTATION	6.01	76.93	58.17

Evidence efficiency. Table 3 explains why SEARCH and READ are separate tools. Raw tree search has high coverage (87.45 All-Hit) but is intentionally broad, returning 16.65 pages on average. This is useful for recall but too expensive to pass directly to the policy. The full reading trajectory inspects far fewer pages (5.79) while achieving the best page-level F1 (58.99), showing that READ converts high-recall candidates into a compact evidence set. ColQwen shows the opposite tradeoff: larger top-k raises All-Hit but lowers F1 by retrieving many extra pages. Tree annotations also matter: removing them drops raw-search All-Hit from 87.45 to 80.05 and page F1 from 38.33 to 29.40, so the mutable tree improves later search rather than only storing notes for final synthesis.

Table 5: Results beyond document understanding. SWE-QA [28] reports five answer-quality dimensions, each scored out of 20, and their overall score out of 100. GAIA [23] reports accuracy (%).

Benchmark	Metric	Direct LLM	DocAtlas
SWE-QA	Correctness	12.25	**15.35**
	Completeness	11.51	**12.85**
	Relevance	16.16	**18.03**
	Clarity	16.92	**17.29**
	Reasoning	14.86	**15.74**
	Overall	71.70	**79.26**
GAIA	Accuracy (%)	60.9	**69.6**

Auxiliary robustness. Table 4 varies the auxiliary model used by SEARCH and REVIEW while holding the RL policy fixed. Replacing GPT-5.4 with Qwen3.5-35B-A3B changes accuracy by only 0.3–0.4 points. The fully open-weight "Self" setting reaches 62.0/63.2 for the 4B/9B policies, within 2 points of the reference. The auxiliary handles tree navigation and note selection, while the trained policy controls tool invocation, page reading, and final answering.

Table 4: Auxiliary-model sensitivity on MMLongBench-Doc with a fixed RL policy. "Self" reuses the policy model for SEARCH/REVIEW; Δ is relative to GPT-5.4.

Aux.	4B	Δ	9B	Δ
GPT-5.4	63.7	0.0	64.4	0.0
Qwen3.5-35B-A3B	63.4	-0.3	64.0	-0.4
Self	62.0	-1.7	63.2	-1.2

4.5 Results Beyond Document Understanding

We evaluate DocAtlas on repository-level code question answering with SWE-QA [28] and general-assistant tasks with GAIA [23]. Both direct answering and DocAtlas use GPT-5.4, which is also used for LLM-based judging. Execution and answer verification follow the official benchmark scripts. Table 5 reports the results.

DocAtlas improves the SWE-QA overall score from 71.70 to 79.26, a gain of 7.56 points, with improvements in all five dimensions. The largest gains are in correctness ($+3.10$), relevance ($+1.87$), and completeness ($+1.34$), showing a consistent improvement in answer content. On GAIA, accuracy increases from 60.9% to 69.6%, a gain of 8.7 percentage points. These results extend the gains from long-document understanding to code question answering and general-assistant tasks.

5 Conclusion

DocAtlas treats long-document understanding as mutable-state interaction: agents search a visually informed tree, read pages, write grounded notes, and review prior findings. This turns a long document into state that the policy updates while it works. Across three benchmarks, DocAtlas improves large VLM agents and helps compact Qwen policies learn tool use through RL. The gains come from multimodal reading, decoupled SEARCH/READ, mutable notes, tree annotations, and SEARCH/REVIEW auxiliaries that can be replaced by open-weight or self models. Additional results on SWE-QA and GAIA show gains in repository-level question answering and general-assistant tasks. Together, the evaluations show the value of an interface that lets a model find, remember, and use evidence.

References

[1] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *ICLR*, 2023.

[2] Longfeng Chen, Zheng Xiao, Juyuan Wang, Zeyu Huang, Yawen Zeng, and Jin Xu. Hear: A holistic extraction and agentic reasoning framework for document understanding. In *ACM MM*, pages 14376–14382, 2025.

[3] 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. *arXiv preprint arXiv:2411.04952*, 2024.

[4] Florin Cuconasu, Giovanni Trappolini, Federico Siciliano, Simone Filice, Cesare Campagnano, Yoelle Maarek, Nicola Tonellotto, and Fabrizio Silvestri. The power of noise: Redefining retrieval for rag systems. In *SIGIR*, pages 719–729, 2024.

[5] Chao Deng, Jiale Yuan, Pi Bu, Peijie Wang, Zhong-Zhi Li, Jian Xu, Xiao-Hui Li, Yuan Gao, Jun Song, Bo Zheng, et al. Longdocurl: a comprehensive multimodal long document benchmark integrating understanding, reasoning, and locating. In *ACL*, pages 1135–1159, 2025.

[6] Yihao Ding, Siwen Luo, Yue Dai, Yanbei Jiang, Zechuan Li, Geoffrey Martin, and Yifan Peng. A survey on mllm-based visually rich document understanding: Methods, challenges, and emerging trends. *arXiv preprint arXiv:2507.09861*, 2025.

[7] Manuel Faysse, Hugues Sibille, Tony Wu, Bilel Omrani, Gautier Viaud, Céline Hudelot, and Pierre Colombo. Colpali: Efficient document retrieval with vision language models. In *ICLR*, 2025.

[8] 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. *arXiv preprint arXiv:2312.10997*, 2023.

[9] Luís F Gomes, Xin Zhou, David Lo, and Rui Abreu. Visdocsketcher: Towards scalable visual documentation with agentic systems. *arXiv preprint arXiv:2509.11942*, 2025.

[10] Ziyu Gong, Chengcheng Mai, and Yihua Huang. Mhier-rag: Multi-modal rag for visual-rich document question-answering via hierarchical and multi-granularity reasoning. *arXiv preprint arXiv:2508.00579*, 2025.

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

[12] Anwen Hu, Haiyang Xu, Liang Zhang, Jiabo Ye, Ming Yan, Ji Zhang, Qin Jin, Fei Huang, and Jingren Zhou. mplug-docowl2: High-resolution compressing for ocr-free multi-page document understanding. In *ACL*, pages 5817–5834, 2025.

[13] Chelsi Jain, Yiran Wu, Yifan Zeng, Jiale Liu, Shengyu Dai, Zhenwen Shao, Qingyun Wu, and Huazheng Wang. Simpledoc: Multi-modal document understanding with dual-cue page retrieval and iterative refinement. In *EMNLP*, pages 28398–28415, 2025.

[14] Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-harness: End-to-end optimization of model harnesses. *arXiv preprint arXiv:2603.28052*, 2026.

[15] Zhanli Li, Huiwen Tian, Lvzhou Luo, Yixuan Cao, and Ping Luo. Deepread: Document structure-aware reasoning to enhance agentic search. *arXiv preprint arXiv:2602.05014*, 2026.

[16] Keliang Liu, Zizhi Chen, Mingcheng Li, Jingqun Tang, Dingkang Yang, and Lihua Zhang. Resolving evidence sparsity: Agentic context engineering for long-document understanding. *arXiv preprint arXiv:2511.22850*, 2025.

[17] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Trans. Assoc. Comput. Linguistics*, 12:157–173, 2024.

[18] Yuliang Liu, Biao Yang, Qiang Liu, Zhang Li, Zhiyin Ma, Shuo Zhang, and Xiang Bai. Textmonkey: An ocr-free large multimodal model for understanding document. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2026.

[19] Xinghua Lou, Miguel Lázaro-Gredilla, Antoine Dedieu, Carter Wendelken, Wolfgang Lehrach, and Kevin P Murphy. Autoharness: improving llm agents by automatically synthesizing a code harness. *arXiv preprint arXiv:2603.03329*, 2026.

[20] Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. Query rewriting in retrieval-augmented large language models. In *EMNLP*, pages 5303–5315, 2023.

[21] 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 *NeurIPS*, BC, Canada, 2024.

[22] Quentin Macé, Antônio Loison, and Manuel Faysse. Vidore benchmark v2: Raising the bar for visual retrieval. *arXiv preprint arXiv:2505.17166*, 2025.

[23] Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: a benchmark for general AI assistants. *arXiv preprint arXiv:2311.12983*, 2023.

[24] Davide Napolitano, Lorenzo Vaiani, and Luca Cagliero. On leveraging multi-page element relations in visually-rich documents. In *COMPSAC*, pages 360–365. IEEE, 2024.

[25] Junbo Niu, Zheng Liu, Zhuangcheng Gu, Bin Wang, Linke Ouyang, Zhiyuan Zhao, Tao Chu, Tianyao He, Fan Wu, Qintong Zhang, et al. Mineru2.5: A decoupled vision-language model for efficient high-resolution document parsing. In *ACL*, 2025.

[26] OpenAI. Introducing gpt-5.4. `https://openai.com/index/introducing-gpt-5-4/`, 2026.

[27] Radek Osmulski, Gabriel de Souza P. Moreira, Ronay Ak, Mengyao Xu, Benedikt Schifferer, and Even Oldridge. Miracl-vision: A large, multilingual, visual document retrieval benchmark. *arXiv preprint arXiv:2505.11651*, 2025.

[28] Weihan Peng, Yuling Shi, Yuhang Wang, Xinyun Zhang, Beijun Shen, and Xiaodong Gu. SWE-QA: Can language models answer repository-level code questions? *arXiv preprint arXiv:2509.14635*, 2025.

[29] Qwen Team. Qwen3.5: Towards native multimodal agents. `https://qwen.ai/blog?id=qwen3.5`, 2026.

[30] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[31] Li Sun, Liu He, Shuyue Jia, Yangfan He, and Chenyu You. Docagent: An agentic framework for multi-modal long-context document understanding. In *EMNLP*, pages 17701–17716, 2025.

[32] Hongchen Wei, Yuanzhe Wang, Bei Liu, Yifan Yang, Qi Dai, Ruichun Ma, Kai Qiu, Yunsheng Li, Dongdong Chen, Chong Luo, et al. Xl-docbench: Benchmarking evidence-grounded extra-long document understanding. *arXiv preprint arXiv:2608.00036*, 2026.

[33] Junda Wu, Yu Xia, Tong Yu, Xiang Chen, Sai Sree Harsha, Akash V. Maharaj, Ruiyi Zhang, Victor S. Bursztyn, Sungchul Kim, Ryan A. Rossi, Julian J. McAuley, Yunyao Li, and Ritwik Sinha. Doc-react: Multi-page heterogeneous document question-answering. In *ACL*, pages 67–78, Vienna, Austria, 2025.

[34] Junyu Xiong, Yonghui Wang, Weichao Zhao, Chenyu Liu, Bing Yin, Wengang Zhou, and Houqiang Li. Docr1: Evidence page-guided grpo for multi-page document understanding. In *AAAI*, volume 40, pages 11178–11186, 2026.

[35] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *ICLR*, Kigali, Rwanda, 2023.

[36] Jiabo Ye, Anwen Hu, Haiyang Xu, Qinghao Ye, Ming Yan, Yuhao Dan, Chenlin Zhao, Guohai Xu, Chenliang Li, Junfeng Tian, et al. mplug-docowl: Modularized multimodal large language model for document understanding. *arXiv preprint arXiv:2307.02499*, 2023.

[37] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

[38] 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. *arXiv preprint arXiv:2410.10594*, 2024.

[39] Xinlei Yu, Chengming Xu, Zhangquan Chen, Yudong Zhang, Shilin Lu, Cheng Yang, Jiangning Zhang, Shuicheng Yan, and Xiaobin Hu. Visual document understanding and reasoning: A multi-agent collaboration framework with agent-wise adaptive test-time scaling. *arXiv preprint arXiv:2508.03404*, 2025.

[40] Qintong Zhang, Xinjie Lv, Jialong Wu, Baixuan Li, Zhengwei Tao, Guochen Yan, Huanyao Zhang, Bin Wang, Jiahao Xu, Haitao Mi, et al. Docdancer: Towards agentic document-grounded information seeking. *arXiv preprint arXiv:2601.05163*, 2026.

[41] Suifeng Zhao, Zhuoran Jin, Sujian Li, and Jun Gao. Finragbench-v: A benchmark for multi-modal rag with visual citation in the financial domain. In *EMNLP*, pages 4215–4249, 2025.

[42] Yuanlei Zheng, Pei Fu, Hang Li, Ziyang Wang, Yuyi Zhang, Wenyu Ruan, Xiaojin Zhang, Zhongyu Wei, Zhenbo Luo, Jian Luan, et al. Doc-v*: Coarse-to-fine interactive visual reasoning for multi-page document vqa. *arXiv preprint arXiv:2604.13731*, 2026.

[43] Dawei Zhu, Rui Meng, Jiefeng Chen, Sujian Li, Tomas Pfister, and Jinsung Yoon. Doclens: A tool-augmented multi-agent framework for long visual document understanding. *arXiv preprint arXiv:2511.11552*, 2025.

A Additional Experimental Results

A.1 XL-DocBench with GPT-5.4

Table 6 reports DocAtlas with GPT-5.4 on XL-DocBench [32]. It achieves 52.47% overall accuracy, 47.56 F1, and 42.91 ANLS.

Table 6: GPT-5.4 comparisons on XL-DocBench, including DocAtlas.

		T1 Core		T2 Structural			T3 Advanced							Aggregate		
Model	In.	Cmp.	Ref.	Rnk.	Cov.	Rec.	SDiff	Unans	Tmp	Cmpl	Ctf	Agg	Con	Acc.	F1	ANLS
GPT-5.4	Img	28.9	26.9	20.5	18.7	23.5	20.3	44.9	24.1	26.2	22.4	23.4	28.6	26.9	29.6	25.9
GPT-5.4	OCR	31.5	36.8	21.0	30.8	40.0	33.9	**70.6**	36.2	32.8	31.0	27.7	23.8	37.3	39.5	34.1
MDocAgent	Agent	43.8	36.3	33.5	30.8	35.7	26.3	22.0	46.6	39.3	48.3	48.9	38.1	35.0	35.8	28.4
SimpleDoc	Agent	48.5	41.0	35.8	28.0	41.7	36.4	60.7	46.6	49.2	55.2	51.1	47.6	44.0	43.7	35.7
DeepRead	Agent	35.3	37.6	27.8	29.1	38.3	32.2	27.6	29.3	36.1	25.9	29.8	33.3	32.2	33.2	26.8
DocAtlas	Agent	**55.36**	**50.72**	**49.16**	**37.79**	**53.65**	**46.59**	64.15	**58.93**	**54.21**	**59.38**	**55.32**	**51.25**	**52.47**	**47.56**	**42.91**

A.2 Full Ablation Results

Table 7 reports the category-level ablation results corresponding to Figure 3.

Table 7: Ablation study of DocAtlas with GPT-5.4 as the agent, SEARCH, and REVIEW LLM on MMLongBench-Doc. Each row removes one component from the full system. Evidence-source columns are text (TXT), layout (LAY), chart (CHA), table (TAB), figure (FIG), and unanswerable (UNA); Acc is overall accuracy and F1 is overall F1.

Variant	TXT	LAY	CHA	TAB	FIG	UNA	Acc	F1
w/o READ	63.9	67.4	63.5	71.9	62.8	69.1	67.2	66.1
w/o NOTE	65.8	72.5	65.3	71.8	63.4	67.9	68.0	66.9
w/o REVIEW	64.9	71.7	62.9	71.3	62.5	71.8	67.9	66.4
w/o layout image	64.1	62.3	63.1	71.6	58.8	67.4	65.7	64.9
w/o sub-image	62.0	70.0	62.9	73.4	63.3	68.6	67.2	65.8
w/o evidence field	64.8	71.6	62.4	70.1	64.2	69.4	67.8	66.8
w/o plan field	65.9	66.7	63.8	70.8	65.6	68.8	68.1	67.1
w/o tree annotation	64.7	71.6	64.0	72.5	63.9	68.6	68.3	67.4
Full DocAtlas	**68.8**	**69.0**	**68.2**	**74.1**	**69.8**	**68.6**	**71.4**	**70.3**

A.3 Full Tool-Usage Statistics

Table 8 reports the full tool-call statistics underlying Figure 3.

Table 8: Average tool calls per question on MMLongBench-Doc. Total sums the calls to the four tools: SEARCH, READ, NOTE, and REVIEW.

Policy	Search	Read	Note	Review	Total
DocAtlas-Plus (GPT-5.4)	1.23	2.81	1.26	0.20	5.50
DocAtlas-4B (Qwen3.5-4B)	1.86	1.45	0.54	0.05	3.90
DocAtlas-4B-RL	1.34	1.86	0.90	0.21	4.31

Table 9 summarizes the relative change induced by RL for the 4B policy. RL increases the mean tool-call count by only 10.5%, but clearly changes the tool mix: SEARCH decreases while READ, NOTE, and REVIEW increase.

Table 9: Tool-use shift from DocAtlas-4B to DocAtlas-4B-RL on MMLongBench-Doc.

Tool	4B no-RL	4B-RL	Abs. change	Rel. change
SEARCH	1.86	1.34	-0.52	-28.0%
READ	1.45	1.86	+0.41	+28.3%
NOTE	0.54	0.90	+0.36	+66.7%
REVIEW	0.05	0.21	+0.16	+320.0%
Total	3.90	4.31	+0.41	+10.5%

A.4 Algorithm Flow

The pseudocode below gives a compact view of one DocAtlas episode. The same loop is used for inference and for RL rollouts; RL additionally stores the completed trajectory and scores it with the reward in Appendix A.5. The pseudocode abstracts away implementation-specific tool schemas, but matches the state variables and tools defined in Section 3.

Algorithm 1 DocAtlas mutable-state interaction loop.

Require: document $D = \{d_1, \ldots, d_N\}$, question q, initial tree G^0, frozen tools SEARCH, READ, NOTE, REVIEW, policy π_θ, maximum turns $T_{\max}$

Initialize $G^1 \leftarrow G^0$, $M^1 \leftarrow \emptyset$, and $H^1 \leftarrow \emptyset$
Set history $h_1 \leftarrow [q, \text{overview}(G^1)]$ and trajectory $\tau \leftarrow [q]$
for $t = 1$ **to** $T_{\max}$ **do**
  $u_t \leftarrow \pi_\theta(h_t)$
  if $u_t = \text{FINISH}(a)$ **then**
    return answer a and trajectory $\tau \parallel (u_t, a)$
  else if $u_t = \text{SEARCH}(z)$ **then**
    $o_t \leftarrow \text{SEARCH}(z; G^t, H^t)$
    $(G^{t+1}, M^{t+1}, H^{t+1}) \leftarrow (G^t, M^t, H^t)$
  else if $u_t = \text{READ}(P)$ **then**
    $o_t \leftarrow \text{READ}(P)$
    $(G^{t+1}, M^{t+1}, H^{t+1}) \leftarrow (G^t, M^t, H^t \cup P)$
  else if $u_t = \text{NOTE}(b, e, p)$ **then**
    $M^{t+1} \leftarrow M^t \cup \{(b, e, p)\}$; $G^{t+1} \leftarrow \text{Enrich}(G^t, e)$; $H^{t+1} \leftarrow H^t$
    $o_t \leftarrow$ acknowledgement
    if archival is enabled **then**
      Replace bulky prior READ observations with short placeholders
    end if
  else if $u_t = \text{REVIEW}(z)$ **then**
    $C_t \leftarrow \text{Cards}(M^t)$; $o_t \leftarrow \text{REVIEW}(z, C_t)$
    $(G^{t+1}, M^{t+1}, H^{t+1}) \leftarrow (G^t, M^t, H^t)$
  else
    $o_t \leftarrow$ invalid-tool message
    $(G^{t+1}, M^{t+1}, H^{t+1}) \leftarrow (G^t, M^t, H^t)$
  end if
  $h_{t+1} \leftarrow h_t \parallel (u_t, o_t)$; $\tau \leftarrow \tau \parallel (u_t, o_t)$
end for
return timeout answer and trajectory τ

RL scoring. For a completed trajectory, set $r = 0$ if the final answer is not boxed; otherwise extract the boxed content and set
$r = \text{Score}_{\text{LongDocURL}}\big(\text{ExtractBoxed}(a), a^\star\big)$.

A.5 Training and Reward Details

Table 10 summarizes the RL configuration used for the compact Qwen3.5 policies. We use the same tool environment as in inference and train only the VLM policy. The tree builder, tool implementations, and auxiliary SEARCH/REVIEW models remain fixed, while the tree and note-store states evolve within each rollout. The implementation uses verl with vLLM multi-turn rollout and FSDP2 actor/reference workers.

Table 10: RL training configuration for DocAtlas compact-policy training. Values reflect the default DAPO run used for the Qwen3.5-4B results unless stated otherwise.

Category	Setting
Base policy	Qwen3.5 VLM initialized from the local checkpoint
Algorithm	DAPO-style GRPO; group-normalized advantages
Rollouts	n=8 sampled trajectories per prompt
Epochs and batch size	1 epoch; train batch size 8; 8 GPUs per node
Actor learning rate	1×10^{-6}
KL and entropy	token-level KL coefficient 0.01; low-variance KL; entropy coefficient 0
Clipping	asymmetric DAPO clipping with ϵ_{low}=0.1, ϵ_{high}=0.3
Sequence limits	max prompt length 8192; max response length 16384
Rollout engine	vLLM; tensor parallel size 1; GPU memory utilization 0.85
Software stack	Python 3.12; PyTorch 2.10; vLLM 0.18; Transformers $\geq$5.5; Ray $\geq$2.54; flash-attn 2.8

Reward function. For RL training, the reward is computed from the final generated answer. The last visible answer is the text after the last `</think>` marker, if present. If this answer does not contain a parsable `\boxed{}` field, the reward is 0. Otherwise, we extract the content inside the final boxed answer and apply the official LongDocURL type-aware evaluation score:

$$r = \text{Score}_{\text{LongDocURL}}\big(\text{ExtractBoxed}(a), a^{\star}\big). \tag{7}$$

The score is computed only from the extracted boxed answer and the ground-truth answer; no separate process term is used.

A.6 Qualitative Trajectory Examples

Figures 4 and 5 show two complete DocAtlas reasoning trajectories. These examples illustrate the full interaction loop: the policy first uses SEARCH to localize a broad candidate region in the mutable document tree, then calls READ on selected pages or visual regions, writes source-grounded NOTE entries, optionally uses REVIEW to recall earlier findings, and finally produces a page-grounded answer.

A.7 DocAtlas Prompt Design

This appendix summarizes the prompt material and tool-call contracts used by DocAtlas. The runtime system prompt is assembled from a fixed preamble, tool-use protocol, optional memory and tree-annotation blocks, the faithfulness rule, the final-answer format, and the four skill descriptions.

A.7.1 System prompt template

Document context and role

You are an expert document analysis assistant. Answer user questions by strictly following the tool usage and reasoning protocol. The document has been pre-indexed into a hierarchical tree structure similar to a detailed table of contents. The user message provides a lightweight table of contents with node IDs, titles, and page ranges, but no authoritative section content. The agent must use tools to inspect actual document evidence.

Tool strategy

Search is the coarse filter: it performs structural tree search over the document hierarchy to locate relevant sections or page ranges. Queries should be specific natural-language descriptions rather than short keywords.

Read is the fine filter: it reads selected pages and returns Markdown text, layout images, and optional cropped sub-images. The agent should read only needed pages, normally at most five pages per call.

Note records established findings, source-grounded evidence, and the remaining plan. It acts as a

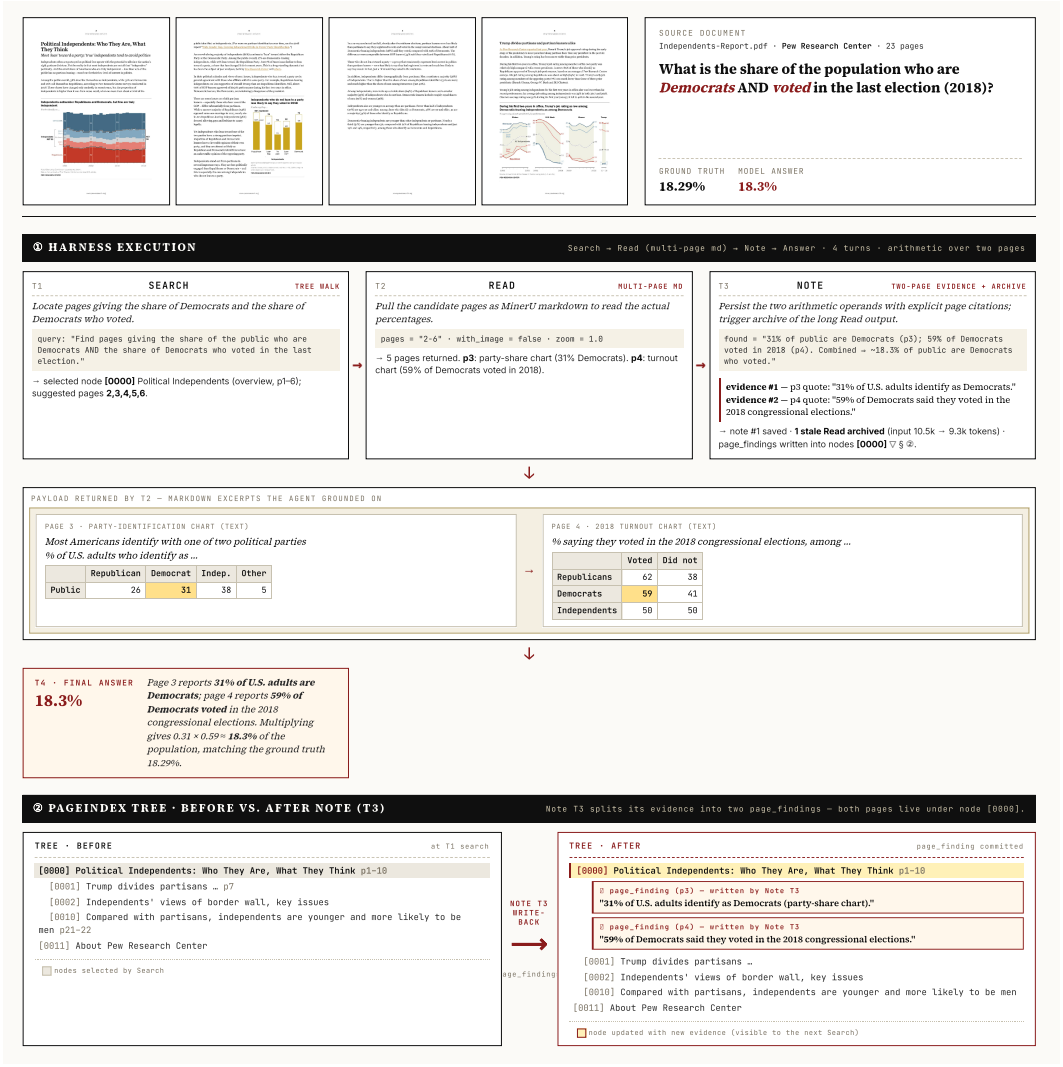


Figure 4: Qualitative trajectory example: multi-hop evidence gathering. DocAtlas searches the document tree, reads selected evidence pages, records the findings in a note, and combines the evidence to answer.

compact reasoning checkpoint.

Review recalls previously saved notes using a focused query. It returns only matched notes rather than a full transcript dump.

Mandatory tool-use protocol

The agent must call at least one tool before answering. The tree overview alone is insufficient; factual answers must be verified by reading page content. For specific questions, the default flow is SEARCH → READ → answer. For broad questions, the agent should search for key sections and read representative pages. When search returns multiple candidates, the agent may start from the most promising subset, but should not skip unread candidates unless it can explain why the candidate set is structurally insufficient. Page numbers in questions are treated cautiously because printed page or slide numbers may differ from physical PDF pages; search is used to resolve this mapping before reading.

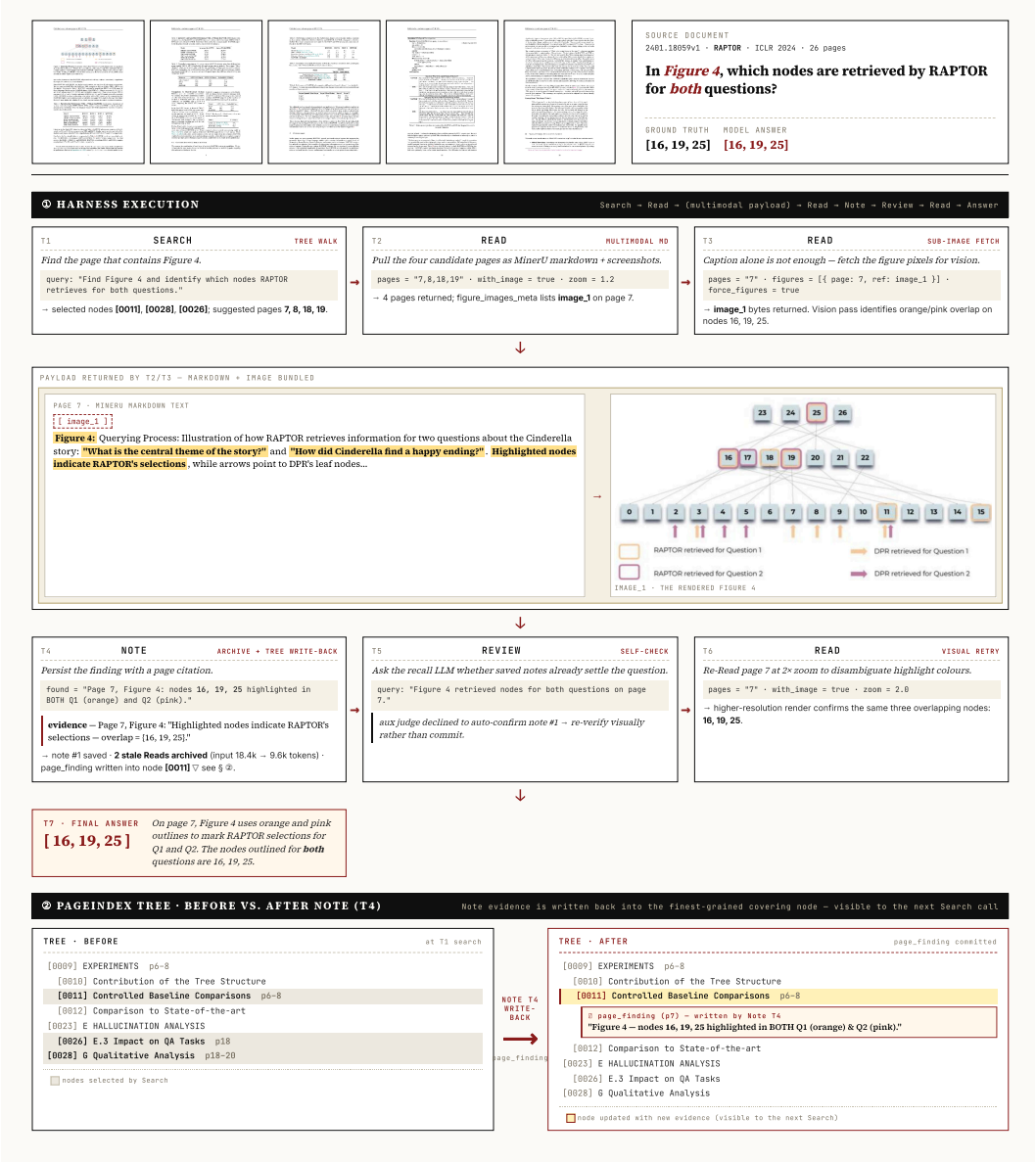


Figure 5: Qualitative trajectory example: complete DocAtlas reasoning flow. The trajectory shows how tree search, selective multimodal reading, source-grounded notes, and final evidence consolidation interact within one episode.

Note and Review protocol

The agent writes a NOTE whenever it finds useful information: confirmed facts, key numbers, page references, negative findings, or changes in search strategy. Every note should contain source-grounded evidence rather than only a high-level summary. For multi-hop or cross-page questions, notes preserve partial findings so later reasoning does not rely on long transcript context.

The agent calls REVIEW when the answer requires combining evidence from multiple notes, when it needs to recall earlier findings, or before answering *Not answerable* after a multi-step search. Review is skipped for simple single-page answers where all evidence is contained in the latest read result.

Memory management and tree annotation

When memory management is enabled, calling NOTE archives earlier bulky READ observations into compact placeholders while preserving source-grounded evidence in the note store. Archived pages can be re-read if needed.

When tree annotation is enabled, notes with explicit page references write page-level findings back into the finest-grained tree nodes covering those pages. These `page_findings` are partial observations from prior query-driven reads, not complete page summaries. Later SEARCH calls can use them as auxiliary hints while still relying primarily on the original tree structure.

Faithfulness and abstention rule

The agent must be faithful to the provided pages. If the read evidence does not contain sufficient information, the final answer must be exactly `Not answerable`. When abstaining, the final-answer line must contain only `Not answerable`; the reasoning may explain why the document lacks the evidence, but must not offer a guess or answer a related question.

Inference-time final-answer format

For benchmark inference, the final response uses:

```
Final answer: <shortest exact span>
Reasoning: <1-3 sentence explanation with page references>
```

The final-answer line should contain only the answer. Numeric answers drop unit words unless required. List answers use a one-line Python list literal and preserve the document wording. Unanswerable cases use exactly `Final answer: Not answerable`.

RL output format. During RL rollouts, the terminal answer is placed in `\boxed{...}` for reward parsing (Appendix A.5), replacing the inference-time final-answer format above. The tool-call contracts remain unchanged.

A.7.2 Per-skill prompt material

Search skill

SEARCH is the discovery step. Given a question, an auxiliary LLM walks the document tree and selects nodes likely to contain the answer. The harness expands selected nodes into suggested page ranges and records search history to avoid repeating the same pages. Search does not read content; it only proposes where to inspect next. Good queries name the target entity, aspect, and constraints, e.g., "Find sections discussing partisan splits on presidential ethics and transparency," rather than a keyword such as "ethics."

Read skill

READ is the only tool that brings document content into the conversation. It fetches selected PDF pages as MinerU Markdown or PyPDF text, with optional full-page screenshots and cropped sub-images. It is used after SEARCH identifies candidate pages, when the actual page text or visual content is needed. The prompt discourages reading more than five pages per call and encourages selective follow-up reads for charts, figures, tables, or scanned pages.

Note skill

NOTE appends a progress-analysis note containing what was found, what remains to do, and evidence linked to page references. The prompt requires exact quoted evidence when possible and discourages unsupported summaries. Notes let the agent preserve important findings without carrying all page observations in active context; later REVIEW calls can retrieve relevant note bodies by query.

Review skill

REVIEW recalls saved notes by query. The harness builds compact note cards, an auxiliary LLM selects relevant cards, and the tool returns the full selected note bodies. Review is used before final answers that combine multiple findings, before a new search when earlier notes may already contain the answer, or before abstaining after a multi-step search. It is not used when no notes exist or when the answer depends only on the latest single read result.

A.7.3 Tool-call schemas

Search schema

Purpose. Locate relevant document-tree nodes and return suggested physical pages.

Field	Type	Description
`query`	string, required	Detailed natural-language search query.

Contract. The query should specify the entity, aspect, and constraints; short keyword queries are discouraged because tree search is LLM-guided rather than lexical matching.

Read schema

Purpose. Fetch page text, page screenshots, or selected sub-images.

Field	Type	Description
`pages`	string, required	Pages or ranges, e.g., `1,3-5,8`.
`with_image`	boolean	Attach full-page screenshots.
`figures`	list	Fetch sub-images by `(page, ref)` from prior metadata.
`force_figures`	boolean	Bypass the minimum-size filter for requested figures.
`zoom`	number	Zoom factor for page screenshots.
`doc_id`	string	Optional document identifier.

Contract. `pages` is the only required field. The prompt recommends no more than five pages per call and uses `figures` only after a previous read exposes a `figure_images_meta` catalog.

Note schema

Purpose. Save a compact, source-grounded reasoning checkpoint and optionally trigger memory/tree side effects.

Field	Type	Description
`found`	string, required	Short summary of established findings.
`plan`	string	Remaining gaps and next intended step.
`evidence`	list	Source-grounded evidence entries.
`side_effect_policy`	enum	Optional archive/enrich behavior override.

Evidence entry.

Field	Type	Description
`type`	enum	`text`, `table`, or `image`.
`source`	string, required	Explicit page reference, e.g., `Page 5`.
`content`	string	Quoted text, table row, or caption.
`filename`	string	Image filename for image evidence.

Contract. Notes should contain page-grounded evidence. When tree annotation is enabled, page references in evidence allow the harness to write findings back into the document tree.

Review schema

Purpose. Recall previously saved notes by query.

Field	Type	Description
`query`	string, required	Focused description of what to recall from notes.

Contract. Review searches only saved notes, not document pages. It returns matched note bodies selected by an auxiliary LLM.

B Broader Impacts, Limitations, and LLM Usage

Broader impacts. DocAtlas is intended to improve access to information in long documents with complex layouts and visual content, such as reports, scientific articles, manuals, filings, and policy documents. Potential positive impacts include reducing the cost of searching and cross-checking evidence in large document collections, improving accessibility for users who need to inspect complex documents, and enabling smaller open-weight VLMs to perform more useful document analysis through tool-mediated interaction instead of relying only on larger backbones. At the same time, the system may be misused to summarize or extract information from sensitive or copyrighted documents without appropriate authorization. It may also produce overconfident answers when retrieved evidence is incomplete or when the source document itself is ambiguous, outdated, or misleading. For high-stakes uses such as legal, medical, financial, or policy decisions, DocAtlas should be treated as an evidence-navigation aid rather than an autonomous decision maker; users should inspect the cited pages and apply domain-specific review.

Limitations. Our evaluation focuses on existing long-document understanding benchmarks and may not cover all document genres, languages, scan qualities, or domain-specific reasoning patterns encountered in deployment. DocAtlas relies on document preprocessing, including Markdown extraction, visual parsing, page-image rendering, cropped sub-images, and a question-agnostic tree. Errors in these preprocessing steps can affect downstream search and reading. The environment also uses auxiliary LLMs for SEARCH and REVIEW. We use GPT-5.4 as a fixed reference auxiliary in the main comparison, and the sensitivity study shows that open-weight and self auxiliaries remain close; still, performance and cost can depend on the quality of these frozen components. The current RL setting trains compact Qwen policies with sparse outcome rewards and pass@16-filtered data from LongDocURL, so further work is needed to understand scaling behavior, reward design, training stability, and transfer to broader domains. Finally, DocAtlas improves evidence access and memory, but it does not guarantee factual correctness: the agent can still miss relevant pages, misread visual elements, write incomplete notes, or combine evidence incorrectly.

Declaration of LLM usage. Large language models are part of the research object studied in this paper: DocAtlas uses VLM agents, auxiliary LLMs for SEARCH and REVIEW, and LLM-as-judge evaluation following the benchmark protocols described in the main text. LLM-based assistance also supported language editing, manuscript organization, and presentation of the authors' experimental results.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the mutable-state harness formulation, the DocAtlas tools, and the main empirical results, which are supported by the method and experiments sections.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Appendix B discusses limitations related to benchmark coverage, preprocessing, auxiliary LLMs, RL training, and remaining factuality risks.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper is empirical and does not present formal theorems or proofs; the equations are used to define the interaction process and RL objective.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The method, tool interface, benchmark protocols, RL data construction, and main evaluation settings are described in Sections 3 and 4 and the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: The code and data are not yet openly released at submission time, although the paper describes the environment, tools, training data construction, and evaluation protocol needed to understand the results.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: Section 4 specifies the evaluated benchmarks, metrics, DocAtlas configuration, RL data construction, reward, optimizer, and key hyperparameters.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: The reported benchmark comparisons and ablations use individual-run scores.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: The paper reports the algorithmic and training setup but does not yet provide a complete per-experiment accounting of GPU type, memory, runtime, or total compute.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work evaluates document-understanding systems on existing benchmarks and discusses limitations, broader impacts, and responsible high-stakes use in Appendix B.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Appendix B discusses positive impacts for evidence access and negative risks such as misuse on sensitive documents and overconfident answers in high-stakes settings.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release a high-risk pretrained language model, image generator, or scraped dataset requiring special safeguards at submission time.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper cites the existing benchmarks, models, document-processing tools, and related systems used for evaluation and comparison.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The new DocAtlas harness, tool contracts, training formulation, and prompt/tool schemas are documented in the method section and Appendix A.7.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The work does not involve crowdsourcing experiments or human-subject studies; it evaluates models on existing document-understanding benchmarks.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: No IRB approval is applicable because the work does not conduct human-subject experiments or collect new human-subject data.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: Appendix B describes the LLMs/VLMs used as agents, auxiliary tools, and evaluators, together with manuscript editing assistance.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.