SimpleDoc: Multi-Modal Document Understanding with Dual-Cue Page Retrieval and Iterative Refinement

Chelsi Jain[1,*], **Yiran Wu**[2,*], **Yifan Zeng**[1,3,*], **Jiale Liu**[2],
Shengyu Dai[4], **Zhenwen Shao**[4], **Qingyun Wu**[2,3], **Huazheng Wang**[1,3]
[1]Oregon State University, [2]Pennsylvania State University,[3]AG2AI, Inc. ,[4]Johnson & Johnson
{jainc, zengyif, huazheng.wang}@oregonstate.edu
{yiran.wu, jiale.liu, qingyun.wu}@psu.edu
{SDai9, ZShao5}@its.jnj.com

Abstract

Document Visual Question Answering (DocVQA) is a practical yet challenging task, which is to ask questions based on documents while referring to multiple pages and different modalities of information, e.g., images and tables. To handle multi-modality, recent methods follow a similar Retrieval Augmented Generation (RAG) pipeline, but utilize Visual Language Models (VLMs) based embedding model to embed and retrieve relevant pages as images, and generate answers with VLMs that can accept an image as input. In this paper, we introduce `SimpleDoc`, a lightweight yet powerful retrieval-augmented framework for DocVQA. It boosts evidence page gathering by first retrieving candidates through embedding similarity and then filtering and re-ranking these candidates based on page summaries. A single VLM-based reasoner agent repeatedly invokes this dual-cue retriever, iteratively pulling fresh pages into a working memory until the question is confidently answered. `SimpleDoc` outperforms previous baselines by 3.2% on average on 4 DocVQA datasets with much fewer pages retrieved. Our code is available at https://github.com/ag2ai/SimpleDoc.

1 Introduction

Documents are a fundamental form for the preservation and exchange of information, and an important source for humans to learn and acquire knowledge (Gu et al., 2021; Chia et al., 2024; Deng et al., 2024). Document question answering is a core task for automated understanding and retrieval of information (Appalaraju et al., 2021; Van Landeghem et al., 2023). Document Visual Question Answering (DocVQA) involves answering questions grounded in multi-modal documents containing text, tables, and images — common in formats like reports and manuals (Suri et al., 2024; Ma et al., 2024b). There are three main challenges in this task: (1) *multiple pages*, where a portion of a long document needs to be processed to answer the question, (2) *multiple references*, where different pages need to be cross-referenced, and (3) *multiple modalities*.

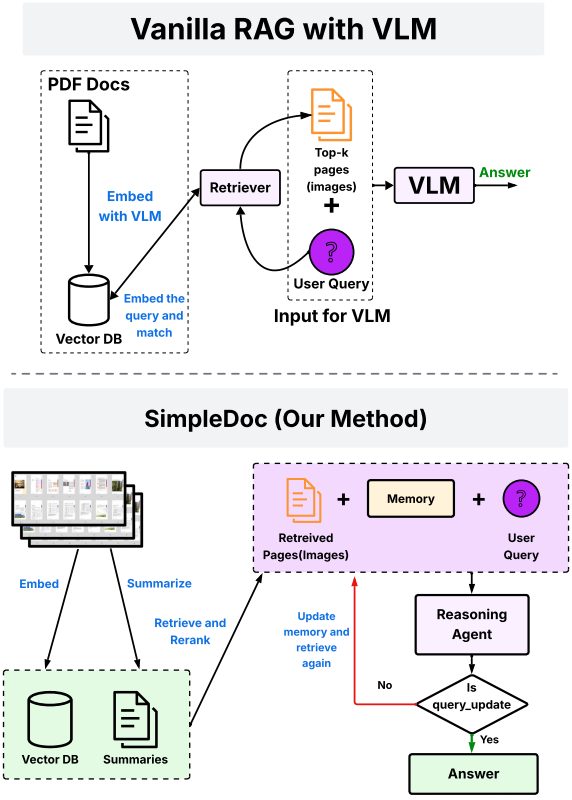

Figure 1: Illustration of the vanilla Retrieval-Augmented Generation (RAG) pipeline and the proposed `SimpleDoc` framework. `SimpleDoc` introduces a two-step page retrieval process that utilizes pre-processed embedding and summaries of each page. During generation, a reasoning agent reviews the retrieved pages and decide whether to give the answer, or produce a new query to retrieve more pages.

Retrieval-augmented generation (RAG) (Lewis et al., 2020) is an effective pipeline to overcome challenges (1) and (2), where relevant information

[*]Equal Contribution.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 28399–28416
November 4-9, 2025

is retrieved by a retrieval model and then fed to a generation model to output the answer. To handle different modalities, several methods have been proposed to pre-process documents by converting different modalities into texts (Memon et al., 2020; Fenniak, 2022; Shinyama et al., 2019). Recently, multi-modal retrieval models such as ColPali (Faysse et al., 2025) are proposed to perform page-level retrieval by treating each page as image (Yu et al., 2024a; Xie et al., 2024). Building on this, M3DocRAG (Cho et al., 2024) proposed a multi-modal RAG system that demonstrated strong performance in DocVQA tasks by combining image and text embeddings for document retrieval. Since multi-agent systems have emerged as an effective method to solve complex tasks and multi-step tasks (Wu et al., 2023; Zheng et al., 2025; Wu et al., 2024), MDocAgent (Han et al., 2025) applied this concept to document QA by designing a multi-agent pipeline composed of dedicated text and image retrieval agents, a critical information extractor, and a final summary agent to collaboratively tackle multi-modal document understanding. Despite MDocAgent's effectiveness, we find it to be overcomplicated and might not utilize the full capacity of recent VLMs.

`SimpleDoc` introduces a simple retrieval augmented framework that leverages modern VLMs without the overhead of complex multi-agent designs. The pipeline unfolds in two stages. First, an offline document-processing stage indexes every page twice: (i) as a dense visual embedding produced by a page-level VLM such as ColPali, and (ii) as a concise, VLM-generated semantic summary that captures the page's most salient content. Second, an online iterative QA stage employs a dual-cue retriever that first shortlists pages via embedding similarity and then asks an LLM, operating solely over the summaries, to decide which of those pages are pertinent to the query and re-rank them by relevance. This ordered subset is handed to a single reasoning agent. The agent reads only the newly selected pages along with a working memory, which preserves important information from previously examined pages, and judges whether the evidence now suffices to answer the question. If it detects missing information, the agent emits a refined follow-up query, prompting another retrieval round and merging the newly distilled notes into memory. This lightweight loop of targeted retrieval and memory-aided reasoning continues until an answer is produced or a preset iteration limit is reached, enabling `SimpleDoc` to flexibly trade retrieval depth for generation quality.

We perform various experiments and analyses to gain an understanding of the VQA problem and to validate the effectiveness of our method. We test on 4 different datasets and find that our method can improve over previous baselines by 3.2 absolute points, with only 3.5 pages retrieved for each question. While the setting of multi-modal, multi-page document-based QA seems new, we find it very much resembles 'traditional' RAG tasks focusing on tasks like HotpotQA (Yang et al., 2018) and 2WIKI (Ho et al., 2020), which usually require retrieved fine-grained chunked texts from given documents. However, M3DocRAG and MDocAgent have had few discussions in this direction. Instead, we do a detailed analysis of these RAG methods and uncover two common strategies: query decomposition and relevant page review. We implement Plan* RAG and Chain-of-note as representations of the common strategies and compare them under the DocVQA setting. To summarize, our contributions are the following:

- We propose `SimpleDoc`, a straightforward and effective framework for multi-modal document question-answering.
- We perform various experiments to test effectiveness of `SimpleDoc`, and analyze and compare with traditional RAG methods in which previous methods on DocVQA are missing.

2 Related Work

Document visual question answering. focuses on answering questions grounded in visual and textual information contained within documents (Ding et al., 2022; Tanaka et al., 2023). Early efforts primarily addressed single-page document images using OCR-based approaches and multi-modal language models (MLMs) (Mathew et al., 2021b,a; Mishra et al., 2019). However, these methods often struggled with the long-context reasoning and complex layouts found in real-world documents. Recently, benchmarks like MP-DocVQA (Tito et al., 2023) and MMLongBench-Doc (Ma et al., 2024b) focus on long multi-page and multi-modal document understanding, posting new challenges to the task (Tanaka et al., 2023). However, recent advances in vision-language models (VLMs) has shown promise for multi-modal document understanding (Liu et al., 2024a, 2023; Chen et al.,

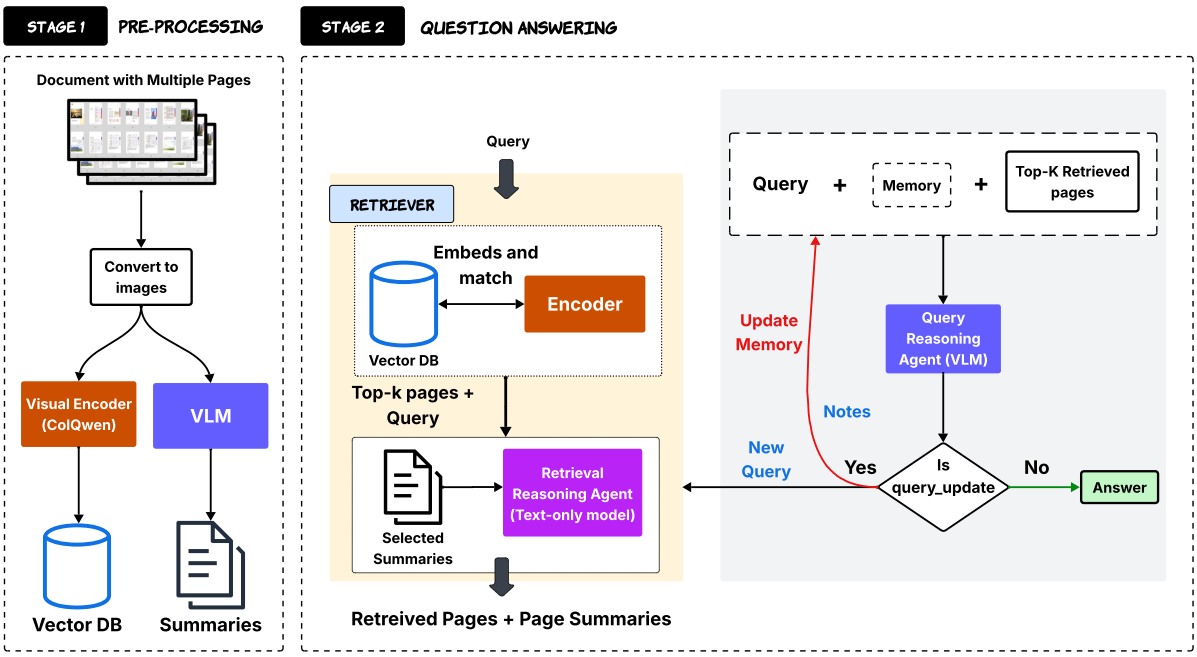

Figure 2: SimpleDoc consists of two stages: (1) offline extraction of visual embeddings and LLM-generated summaries for all document pages, and (2) an online reasoning loop that performs retrieval via embedding and summary-based re-ranking, followed by answer generation with a memory-guided VLM agent that iteratively refines its query if needed.

2022; Bai et al., 2025; Xie et al., 2025; Ma et al., 2024a). ColPali (Faysse et al., 2025) introduces a new concept of treating document pages as images to produce multi-vector embeddings, where pages can be retrieved for each query. Other methods such as VisRAG (Yu et al., 2024a) and VDocRAG (Tanaka et al., 2025) also convert pages as images to avoid missing information from parsing text and image separately from one page. From ColPali, M3DocRAG (Cho et al., 2024) proposed a multi-modal RAG pipeline that retrieves relevant document pages across large document corpora and feeds them into a vision language model. MDocAgent (Han et al., 2025) extended this by introducing specialized agents for handling cross-modal retrieval and reasoning over long documents.

Retrieval augmented generation (RAG) has become a powerful strategy for knowledge-intensive tasks by supplementing language models with external context, which consists of two core steps: retrieve and generate (Jiang et al., 2023a; Gao et al., 2023). Many works have been proposed to improve RAG, such as training effective embedding models (Karpukhin et al., 2020; Khattab and Zaharia, 2020a), query rewrite and decomposition (Ma et al., 2023; Peng et al., 2024; Chan et al., 2024; Verma et al., 2025; Lee et al., 2024; Wang et al., 2024), constructing different forms of databases (e.g., knowledge graphs) (Gaur et al., 2022; Edge et al., 2024; Liu et al., 2025), improving quality of retrieved context (Yu et al., 2024b; Chen et al., 2024), augmenting the RAG process (Asai et al., 2023; Trivedi et al., 2022a; Liu et al., 2024b), and many others (Jiang et al., 2023b). Most of the RAG methods focus on knowledge and reasoning tasks that only require text-based retrieval (e.g., HotpotQA) (Yang et al., 2018; Geva et al., 2021; Trivedi et al., 2022b; Mallen et al., 2023; Ho et al., 2020; Kwiatkowski et al., 2019). While we are targeting the Document Visual understanding task, we find that many core ideas might also be effective in DocVQA. Thus, we also implement and test two RAG methods: Chain-of-Notes (Yu et al., 2024b), which improves retrieval context for better generation, and Plan*RAG (Verma et al., 2025), which decomposes queries and augments the generation process for better retrieval, to help understand how previous methods can be used on DocVQA.

3 Method

Below we introduce SimpleDoc, an effective framework for DocVQA. SimpleDoc consists of two stages: an offline document processing phase followed by an online iterative retrieval-augmented question answering phase. Our framework features

the following: 1. Enhanced page retrieval through a combination of vector and semantic representations. 2. Continuous refinement via iterative retrieval and memory update. Figure 2 illustrates the overall pipeline of our approach.

3.1 Offline Document Processing

The initial stage involves pre-processing and indexing each document to create a searchable representation. We treat each page as a unit, and use two VLMs to get both vector and semantic representations of each page. For vector embedding, we employ VLM like ColPali (Faysse et al., 2025) that are trained to generate embeddings for document pages. For semantic representation, we use a general VLM guided by a predefined prompt to produce a summary (typically 3-5 sentences) that includes the salient information of that page. These summaries are designed to highlight information that might be generally relevant for answering potential future questions without prior knowledge of any specific user query.

Specifically, given a document D consisting of j pages $D = p_1, p_2, ..., p_j$, we use a vision embedding model to generate embedding vectors $E = \{e_1, e_2, \ldots, e_j\}$ for each page, and use a VLM to generate j summaries $S = \{s_1, s_2, ..., s_j\}$.

3.2 Multi-modal Question Answering

For retrieval, we use a VLM to retrieve pages through embedding similarity, and a VLM to look at the summaries and re-rank those retrieved pages. During the question answering phase, we build a reasoner agent that can automatically decide whether to retrieve more information and iteratively refine its own memory with newly retrieved pages.

Page Retrieval Given a query q and its document D, we first embed the given query and retrieve k pages with the highest MaxSim score (Khattab and Zaharia, 2020b). Then, we pass q and k summaries of the retrieved pages S_k into an LLM (can be text-only) to select and rank the relevant pages. The model returns an ordered list of page indices $C = c_1, c_2, \ldots, c_n$ based on their perceived relevance to the query. Note that the number of relevant pages is automatically and dynamically chosen by the model. Since the re-rank is based on the retrieved pages from embedding, so $n < k$ pages are later sent to the reasoner agent, keeping the input size manageable. In this step, we also ask the LLM to generate an overall document-level summary s_{DOC} that contextualizes the entire document in relation to the current query, serving as the initial working memory of the reasoner agent.

Algorithm 1 `SimpleDoc`

Require: query q, per–page embeddings E and summaries S, cutoff k, max iterations L
Ensure: answer a or failure notice

$q_{\text{cur}} \leftarrow q$
$M \leftarrow \varnothing$
for $\ell \leftarrow 1$ **to** L **do**
  $s_{\text{DOC}}, C \leftarrow \mathit{RetrievePages}(q_{\text{cur}}, E, S, k)$
  $I_C \leftarrow \{\, i_c \mid c \in C\}$; $T_C \leftarrow \{\, t_c \mid c \in C\}$
  $M \leftarrow M \cup s_{\text{DOC}}$
  $(\mathit{is\_solved}, a, m', q') \leftarrow$
    $\textsc{Reasoner}(q, I_C, T_C, M)$
  if *is_solved* **then**
    return a
  else
    $M \leftarrow M \cup \{m'\}$
    $q_{\text{cur}} \leftarrow q'$
return FAIL

Generation We treat the retrieved relevant pages as images, denoted as $I_C = \{i_{c_1}, i_{c_2}, \ldots, i_{c_n}\}$. Those pages are also converted into text, denoted as $T_C = \{t_{c_1}, t_{c_2}, \ldots, t_{c_n}\}$. We input I_C, T_C, input query q and a working memory M (initialized to s_{DOC}) into a reasoner agent (backed by a VLM), and ask it to determine if the question can be solved with the given context.

The reasoner can produce one of three distinct response types:

- **Answer**: If the provided pages contain sufficient information, the reasoner formulates a direct answer to the query.
- **Not Answerable**: If the question cannot be answered by the document.
- **Query Update**: If the reasoner believes the answer exists within the document but on pages not yet retrieved, it outputs a note of current pages m' and generates a new query q' that asks for missing information.

Iterative Refinement Self-reflection has been proven an effective method in LLMs (Shinn et al., 2023; Madaan et al., 2023), and we employ a similar mechanism where the LLM retrieved additional pages as needed. If the reasoner agent decides that it cannot answer after the initial retrieval, an iterative process begins to continue retrieving new

pages. As shown in Algorithm 1, we maintain a memory module M to preserve useful information from previous retrievals. When the reasoner agent outputs a query update, we retrieve new page numbers C' based on the refined query q', update the memory module M with the notes m', and call the reasoner again with the following inputs: $\{q, I_{C'}, T_{C'}, M\}$. The process stops when an answer is produced, the query is marked unanswerable, or a maximum iteration limit L is reached, after which the question is marked "not answerable".

Memory Update Mechanism The memory module maintains a running context throughout the iterative reasoning process:

- An initial **document-level summary** is generated during the first retrieval pass.
- At each iteration, the reasoning agent emits **notes** summarizing what has been found so far and what information is still missing.
- This combined memory is passed as context to all subsequent reasoning rounds and is updated incrementally as part of the agent's output.

This design ensures that the agent can carry forward key evidence, avoid re-reading redundant content, and refine its search trajectory over multiple iterations. We evaluated its effect by disabling memory module; results are in Appendix A.4.

4 Experiments

Our experiment is organized as follows: In Section 4.1, we present the main results of our method and baselines on 4 different datasets. In Section 4.2, we further experiment on MMLongBench using different models. In Section 4.3, we adopt and implement two other RAG methods that were originally proposed for knowledge Question Answering, Finally in Section 4.4, we test variations of `SimpleDoc` and further analyze our method.

4.1 Main Results

Datasets. We evaluate `SimpleDoc` on 4 diverse PDF document understanding benchmarks, providing a robust testbed for assessing performance across varied document types, lengths, and retrieval complexities:

1) MMLongBench (Ma et al., 2024b): This dataset is designed to test document reasoning over long PDFs, containing complex layouts and multimodal components. The dataset contains 1073 questions across 135 documents, with an average length of 47.5 pages per document.

2) LongDocURL (Deng et al., 2024): Another large-scale multi-modal benchmark aimed at evaluating document retrieval and reasoning. It has over 33,000 document pages and includes 2,325 question samples.

3) PaperTab (Hui et al., 2024): It focuses on the extraction and interpretation of the tabular data from the research papers, providing 393 questions from over 307 academic documents.

4) FetaTab (Hui et al., 2024): A table-based question answering dataset using tables extracted from Wikipedia articles. It presents 1,023 natural language questions across 878 documents, requiring models to generate free-form answers.

Baselines. We compare with two baselines: (1) *M3DocRAG* (Cho et al., 2024) which uses an image retrieval model to retrieve top-k pages, and a VLM to generate an answer with retrieved pages. (2) *MDocAgent* (Han et al., 2025) employs both text and image retrieval models to retrieve two sets of pages, then top-k pages from both sets will be used for generation. MDocAgent uses 5 different agents and require both a VLM and a text model. We also include the results of using a VLM to solve the question directly, and results of using VLM with the ground-truth pages included as images (denoted as GT pages), serving as lower and upper bounds.

Metrics. For this experiment, we evaluate model performance with *Binary Correctness (Accuracy)*. We classify each model response as either correct or incorrect and compute the accuracy as the ratio of correct responses to the total number of questions. We use **GPT-4.1** as an automatic evaluator to judge response correctness against ground truth answers and set the temperature to 0.

Implementation Details. We use the same models for `SimpleDoc` and baselines for rigorous comparison. For visual embedding model, we use `ColQwen-2.5` for all methods, which is the latest model trained with ColPali (Faysse et al., 2025)'s strategy (See Table 7 for a comparison with ColPali), and we use `Qwen2.5-VL-32B-Ins` whenever a VLM is needed. For MDocAgent, we use ColBERTv2 (Khattab and Zaharia, 2020a) as the text retrieval model following the original paper, and `Qwen3-30B-A3B` as the text model. For `SimpleDoc`, we use `Qwen2.5-VL-32B-Ins` for per-page summarization during pre-processing. Note that the

Method	Pg. Ret.	MMLongBench	LongDocUrl	PaperTab	FetaTab	Avg. Acc
LVMs						
Qwen2.5-VL-32B-Instruct	–	22.18	19.78	7.12	16.14	16.31
Qwen2.5-VL-32B-Instruct + Ground-Truth pages	–	67.94	30.80	-	-	-
RAG methods (top 2)						
M3DocRAG (Qwen2.5-VL-32B)	2	41.8	50.7	50.1	75.2	54.4
MDocAgent (Qwen3-30B + Qwen2.5-VL-32B)	4	50.6	56.8	50.9	80.3	59.6
RAG methods (top 6)						
M3DocRAG (Qwen2.5-VL-32B)	6	41.8	53.1	60.1	79.8	58.7
MDocAgent (Qwen3-30B + Qwen2.5-VL-32B)	12	55.3	63.2	64.9	**84.5**	66.9
RAG methods (top 10)						
M3DocRAG (Qwen2.5-VL-32B)	10	39.7	52.2	56.7	78.6	56.8
MDocAgent (Qwen3-30B + Qwen2.5-VL-32B)	20	54.8	61.9	63.1	84.1	65.9
Ours (top-10 and top-30)						
SimpleDoc (Qwen3-30B + Qwen2.5-VL-32B)	3.2	59.55	72.26	64.38	80.31	69.12
SimpleDoc (Qwen3-30B + Qwen2.5-VL-32B)	3.5	**60.58**	**72.30**	**65.39**	82.19	**70.12**

Table 1: Accuracy(%) on 4 different DocVQA datasets. We use ColQwen-2.5 as the retrieval model for all methods. *Pg. Ret.* indicates the actual pages used during generation.

Table 2: All-Match Retrieve Rate, and Page-level F-1 Score on MMLongBench (See Section A.3 for calculation). We present the results for ColQwen (used by M3DocRAG and MDocAgent) and our retrieval.

Method	Avg Ret. Pages	All Hit %	F1 Score
ColQwen-2.5	2	64.12	38.75
ColQwen-2.5	6	76.42	24.36
ColQwen-2.5	10	**83.60**	18.38
Ours (top-10)	3.19	65.72	61.42
Ours (top-30)	3.46	67.37	**62.22**

summarization only needs to be performed once. We use `Qwen3-30B-A3B` to for page retrieval. For baselines, we test with top-k set to 2, 6, 10. For our method, we set top-k to 10 and 30 for embedding retrieval. All prompts used in our method is shown in Appendix A.6.

Results Analysis Table 1 shows that `SimpleDoc` achieves the highest average accuracy of 70.12%, outperforming all the baselines with different top-k retrieval settings. On MMLongBench and LongDocURL, which contain long, diverse, and multimodal documents, our method significantly outperforms MDocAgent by +5.3% and +9.1%, respectively, demonstrating strength in addressing complex queries that require aggregating information dispersed across different sections of a document. However, on FetaTab, a heavily table-centric dataset, `SimpleDoc` performs lower than MDocAgent. We attribute this to MDocAgent's explicit multi-agent design, which uses a dedicated image agent to focus on another modality (table grids) and is especially effective for this specific type of table-based QA. In contrast, `SimpleDoc` treats pages as images to feed into a single agent. Thus, `SimpleDoc` is more robust and effective across questions that require diverse evidence types.

Table 1 also lists the average number of pages each system retrieves. `SimpleDoc` needs only 3.5 pages per question yet achieves the best overall accuracy. By contrast, MDocAgent attains 59.6% accuracy when it reads 4 pages, which is about 10 percentage points below our method. Notably, both MDocAgent and M3DocRAG reach their peak accuracy at top-k=6 rather than 10, implying that indiscriminately adding pages can hurt performance. To understand this effect, Table 7 reports two retrieval metrics. 1) The all-hit rate gauges coverage, the fraction of questions for which the entire gold evidence set appears among the retrieved pages. 2) The page-level F1 score captures efficiency, rewarding systems that surface the right pages while avoiding noise. For `ColQwen-2.5`, raising k from 2 to 10 boosts coverage but reduces F1, showing that many of the extra pages are irrelevant. Thus, top-k=6 reflects a better balance between coverage and conciseness, which in turn yields higher answer accuracy for the agent baselines. In contrast, `SimpleDoc` attains nearly the same coverage as `ColQwen-2.5` at k=2 yet more than doubles its F1, demonstrating that our retriever supplies almost all necessary evidence with far less clutter. Overall, `SimpleDoc` delivers the best coverage-versus-conciseness trade-off while avoiding trial-and-error to find the best top-k retrieval

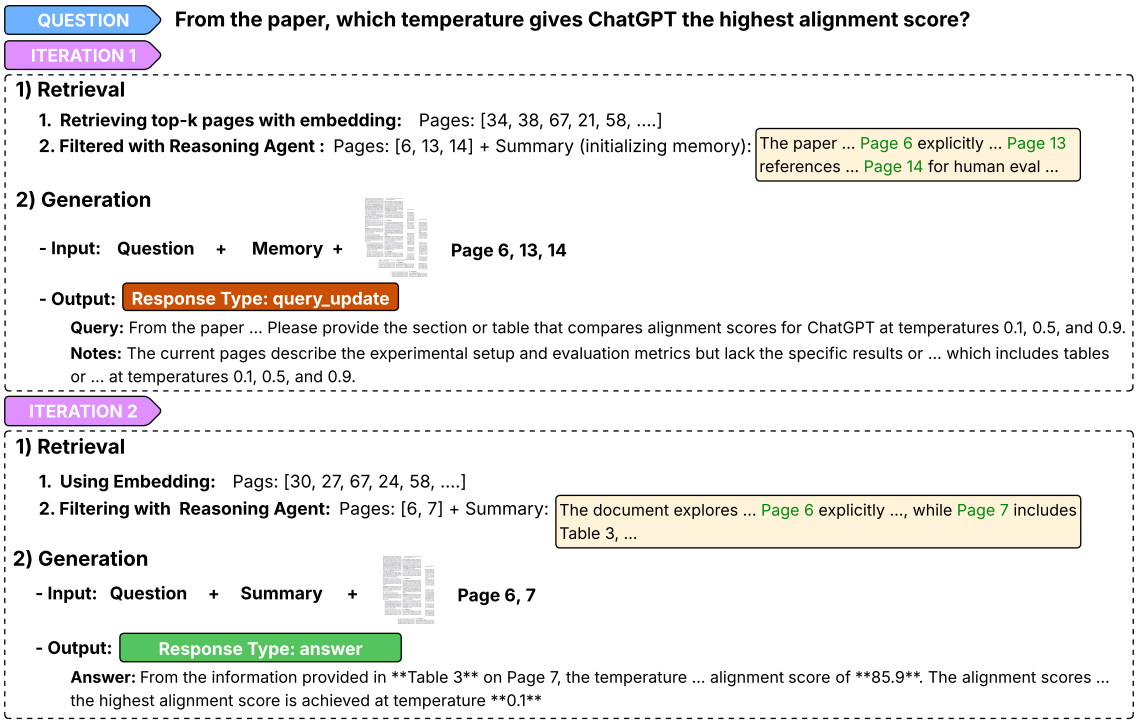

Figure 3: An example run of SimpleDoc's iterative reasoning solving a question. In the first round, the agent retrieves Pages 6, 13, and 14 based on embedding and summary-based filtering. However, the retrieved pages only describe the experimental setup and evaluation metrics without giving exact alignment scores. The agent identifies this gap and generates a refined query asking specifically for a section or table comparing scores at different temperatures. This updated query retrieves Page 7, which contains Table 3 with the required information, allowing the agent to correctly answer that temperature 0.1 yields the highest alignment score (85.9).

numbers, giving the reasoner everything it needs while keeping the reading budget minimal.

Qualitative Analysis As shown in Figure 3, SimpleDoc reasons iteratively. Initially, it retrieves broadly relevant pages but lacking specific details needed to answer the question. Recognizing the gap, the agent refines the query to target missing information, retrieves the precise page containing the relevant table, and answers successfully. This demonstrates how SimpleDoc detects incomplete evidence and adaptively improves retrieval to resolve complex queries.

4.2 Results with different models

In Table 3, we test with smaller models (Qwen2.5-VL-7B-Instruct + Qwen-3-8B) with detailed results on MMLongBench to further validate our method. Note that Qwen-3-8B are text-only models and used in MDocAgent (Text Agent) and our method (for retrieval). Our method outperforms all baselines in terms of avg. accuracy (ACC) for both models. Under the smaller 7B/8B model setting, our method achieves 50% overall accuracy, improving over MDocAgent by +6.62 points, which is a bigger gap compared to using larger models (+4.15 points). When broken down by evidence source, our model achieves the best performance on three out of five modalities. We note that MDocAgent are competitive on charts and tables with specialized agents, which is consistent with our observation and analysis in Section 4.1. When broken down by number of evident pages, our methods have similar results compared with MDocAgent on multi-page (MUL) and single-page (SIN) reasoning with different models. However, SimpleDoc achieves better results on unanswerable questions, used to test hallucinations, showcasing its ability to abstain from guessing when no valid evidence is present.

4.3 Other RAG methods

We also adopt and evaluate two RAG methods that originally focus on knowledge question answering tasks: **(1) Plan*RAG** (Verma et al., 2025): first decomposes a question into sub-queries that form a directional acyclic graph (DAG). It starts with solving the leaf sub-queries, and incorporates the

Method	Evidence Source					Evidence Page			ACC
	TXT	LAY	CHA	TAB	FIG	SIN	MUL	UNA	
Qwen2.5-VL-7B-Instruct + Qwen-3-8B									
VLM + GT pages	51.32	45.38	37.71	40.09	47.83	58.90	35.01	77.97	54.99
M3DocRAG (top-6)	43.21	39.98	36.05	31.60	42.01	55.46	24.78	8.72	35.50
MDocAgent (top-6)	47.04	38.98	**47.09**	**41.04**	39.93	**59.45**	28.57	33.49	43.80
Ours	**49.67**	**42.02**	44.57	37.79	**42.14**	58.69	**31.65**	**62.11**	**50.42**
Qwen2.5-VL-32B-Instruct + Qwen-3-30B-A3B									
VLM + GT pages	63.25	66.39	58.86	65.44	57.53	72.60	55.46	77.53	67.94
M3DocRAG	46.69	41.53	45.35	39.15	43.75	58.61	30.61	22.48	41.80
MDocAgent	57.49	50.00	54.65	**56.13**	**52.78**	68.70	**42.86**	45.41	55.30
Chain-of-Notes†	36.75	35.29	38.46	32.26	33.44	49.59	21.69	50.00	40.45
Plan*RAG†	46.03	36.13	43.75	38.71	37.12	54.88	25.35	23.89	38.58
Ours	**59.93**	**51.26**	**54.86**	51.15	51.17	**70.76**	39.22	**67.40**	**59.55**

Table 3: Performance with different models on **MMLongBench**. We present detailed accuracy for questions with five different evidence sources: text (TXT), layout (LAY), chart (CHA), table (TAB), and figure (FIG); different numbers of evidence pages (single (SIN), multiple (MUL), unanswerable (UNA), and average accuracy. We also test two RAG methods originally proposed for knowledge QAs on MMLongBench (labeled with †).

previous subquery+answer when solving the next queries, until the original question. This features the query-decomposition and augmented process strategies, which are common in RAG methods. **(2) Chain-of-Notes** (Yu et al., 2024b) taking notes of retrieved paragraphs and then using them for more precise generation. We do the following to adapt them to our setting: we use ColQwen2.5 to retrieve document pages, and use VLM for generation, which is the same as other baselines.

Table 3 reports the performance of the two RAG baselines when paired with `Qwen2.5-VL-32B`. Both Chain-of-Note and Plan*RAG underperform methods tailored for DocVQA, showing that directly applying text-oriented RAG techniques is insufficient for this domain. Our analysis highlights potential failure causes: (1) Chain-of-Note relies on page-level image summaries, which can miss fine details such as exact numbers in tables or words in charts and layouts. A single summary per page is often too general, making cross-page reasoning and precise answers difficult, resulting in 40.4% accuracy. (2) Plan*RAG processes full-page images and decomposes the main question into sub-questions via a query graph. However, the generated acyclic graph is frequently inaccurate, leading to off-target sub-queries. Each sub-query retrieves top-k image pages, answers them, and aggregates the results, a multi-step pipeline that adds complexity and propagates errors.

4.4 Additional Analysis of `SimpleDoc`

In this section, we do more experiments to decompose and analyze our method.

Top-k	Avg. Page Used	Acc.
2	2.15	56.66
6	2.75	58.25
10	3.19	59.55
30	3.46	60.58

Table 4: Our method with different top-k numbers for embedding retrieval on MMLongBench. *Avg. Page Used* denotes the actual number of pages seen by the reasoner agent.

Varying top-k for embedding retrieval. In `SimpleDoc`, we first retrieve top-k pages using embeddings, and then re-rank them based on summaries. With retrieval, we can filter and bound

Iteration	1	2	3
Accuracy	58.62	59.27	59.55
# Query Update	182	121	97

Table 5: Performance of `SimpleDoc` on MMLongBench across different iterations, showing accuracy and number of query updates.

the maximum number of pages before re-ranking. We evaluate different k values to examine how the initial candidate set affects retrieval. Larger k provides the agent with more options to recover pages missed by embeddings, but we observed that the agent did not select significantly more pages even when k was large, indicating it dynamically filters for truly relevant content.

Results with different iterations. Table 5 illustrates the benefits of our iterative refinement strategy on MMLongBench. The observed trend shows that additional iterations allow `SimpleDoc` to progressively enhance understanding and locate crucial information initially missed. This targeted re-querying leads to improved accuracy, while the decreasing number of query updates indicates the system is either satisfying the information need or recognizing when an answer cannot be found within the document.

We performed an **ablation on the dual-cue retriever**, showing that removing the summary-based re-ranking stage and relying only on embeddings causes a substantial drop in QA accuracy and retrieval F1. To quantify the contribution of the dual-cue retrieval mechanism, we performed an ablation removing the summary-based filtering stage and relying solely on embedding retrieval.

Setting	QA Accuracy
Dual retrieval (Embedding + Summaries)	59.55
Single retrieval (Embedding only)	54.80

Table 6: Effect of removing summary-based re-ranking on MMLongBench.

Disabling the summaries leads to a drop in final QA Accuracy to 54.80% and reduces retrieval F1 from 61.42% (Table 2) to 23.32%. We also observe a higher incidence of false positive retrievals, where non-relevant pages are passed to the reasoner, frequently causing hallucinated answers. These results demonstrate that summary-based re-ranking significantly improves retrieval precision and overall answer quality.

We also measured **computational statistics** in terms of token-level input/output to evaluate efficiency across model variants and compared SimpleDoc with MDocAgent (Appendix A.4). Furthermore, we conducted an **error analysis** categorizing the main failure cases into eight types: retrieval failure, partial evidence retrieval, hallucination, long-context overload, multi-modal misalignment, ambiguous question interpretation, unanswerable questions, and layout errors, with examples shown in Appendix A.5.

5 Conclusion

We present `SimpleDoc`, an effective framework for multi-modal document QA. `SimpleDoc` consists of an efficient retrieve module that utilizes both dense-vector embedding and summary, to retrieve the pages efficiently, and a reasoning agent that can detect and remedy missing evidence iteratively. Empirical results across 4 DocVQA benchmarks confirm that `SimpleDoc` surpasses prior RAG-style systems and multi-agent baselines with fewer components and fewer page retrievals. These results highlight how modern VLMs can be used on retrieval-augmented multi-modal reasoning.

6 Limitations

In this work, we only experiment with single-document VQAs, while the embedding retrieval method can be readily extensible to retrieve from the whole document database. We believe there are still many interesting research questions under this scenario. We focus on test-time scaling methods instead of training, and we think more RAG methods that require training (Asai et al., 2023; Chan et al., 2024) can be utilized for this task. Finally, graph-based database and retrieval methods are also future directions to explore(Edge et al., 2024; Liu et al., 2025).

References

Srikar Appalaraju, Bhavan Jasani, Bhargava Urala Kota, Yusheng Xie, and R Manmatha. 2021. Docformer: End-to-end transformer for document understanding. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 993–1003.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *Preprint*, arXiv:2310.11511.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, and 1 others. 2025. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*.

Chi-Min Chan, Chunpu Xu, Ruibin Yuan, Hongyin Luo, Wei Xue, Yike Guo, and Jie Fu. 2024. Rq-rag: Learning to refine queries for retrieval augmented generation. *Preprint*, arXiv:2404.00610.

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

Wenhu Chen, Hexiang Hu, Xi Chen, Pat Verga, and William W Cohen. 2022. Murag: Multimodal retrieval-augmented generator for open question answering over images and text. *arXiv preprint arXiv:2210.02928*.

Yew Ken Chia, Liying Cheng, Hou Pong Chan, Chaoqun Liu, Maojia Song, Sharifah Mahani Aljunied, Soujanya Poria, and Lidong Bing. 2024. M-longdoc: A benchmark for multimodal super-long document understanding and a retrieval-aware tuning framework. *arXiv preprint arXiv:2411.06176*.

Jaemin Cho, Debanjan Mahata, Ozan Irsoy, Yujie He, and Mohit Bansal. 2024. M3docrag: Multi-modal retrieval is what you need for multi-page multi-document understanding. *Preprint*, arXiv: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. 2024. Longdocurl: a comprehensive multimodal long document benchmark integrating understanding, reasoning, and locating. *Preprint*, arXiv:2412.18424.

Yihao Ding, Zhe Huang, Runlin Wang, YanHang Zhang, Xianru Chen, Yuzhong Ma, Hyunsuk Chung, and Soyeon Caren Han. 2022. V-doc: Visual questions answers with documents. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 21492–21498.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Manuel Faysse, Hugues Sibille, Tony Wu, Bilel Omrani, Gautier Viaud, Céline Hudelot, and Pierre Colombo. 2025. Colpali: Efficient document retrieval with vision language models. *Preprint*, arXiv:2407.01449.

Mathieu Fenniak. 2022. The PyPDF2 library. Version 2, authors including PyPDF2 Contributors.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2:1.

Manas Gaur, Kalpa Gunaratna, Vijay Srinivasan, and Hongxia Jin. 2022. Iseeq: Information seeking question generation using dynamic meta-information retrieval and knowledge graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 10672–10680.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *arXiv preprint arXiv:2101.02235*.

Jiuxiang Gu, Jason Kuen, Vlad I Morariu, Handong Zhao, Rajiv Jain, Nikolaos Barmpalios, Ani Nenkova, and Tong Sun. 2021. Unidoc: Unified pretraining framework for document understanding. *Advances in Neural Information Processing Systems*, 34:39–50.

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

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of COLING 2020*. 2WikiMultiHopQA dataset introduced.

Yulong Hui, Yao Lu, and Huanchen Zhang. 2024. Uda: A benchmark suite for retrieval augmented generation in real-world document analysis. *Preprint*, arXiv:2406.15187.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023a. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7969–7992.

Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023b. Active retrieval augmented generation. *Preprint*, arXiv:2305.06983.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick SH Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pages 6769–6781.

Omar Khattab and Matei Zaharia. 2020a. Colbert: Efficient and effective passage search via contextualized late interaction over bert. *Preprint*, arXiv:2004.12832.

Omar Khattab and Matei Zaharia. 2020b. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 39–48.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Myeonghwa Lee, Seonho An, and Min-Soo Kim. 2024. Planrag: A plan-then-retrieval augmented generation for generative large language models as decision makers. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6537–6555.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2024a. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26296–26306.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916.

Pei Liu, Xin Liu, Ruoyu Yao, Junming Liu, Siyuan Meng, Ding Wang, and Jun Ma. 2025. Hm-rag: Hierarchical multi-agent multimodal retrieval augmented generation. *arXiv preprint arXiv:2504.12330*.

Yanming Liu, Xinyue Peng, Xuhong Zhang, Weihao Liu, Jianwei Yin, Jiannan Cao, and Tianyu Du. 2024b. Ra-isf: Learning to answer and understand from retrieval augmentation via iterative self-feedback. *arXiv preprint arXiv:2403.06840*.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023. Query rewriting in retrieval-augmented large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5303–5315.

Xueguang Ma, Shengyao Zhuang, Bevan Koopman, Guido Zuccon, Wenhu Chen, and Jimmy Lin. 2024a. Visa: Retrieval augmented generation with visual source attribution. *arXiv preprint arXiv:2412.14457*.

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. 2024b. Mmlongbench-doc: Benchmarking long-context document understanding with visualizations. *Preprint*, arXiv:2407.01523.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*. PopQA dataset introduced.

Minesh Mathew, Viraj Bagal, Rubèn Pérez Tito, Dimosthenis Karatzas, Ernest Valveny, and C. V Jawahar. 2021a. Infographicvqa. *Preprint*, arXiv:2104.12756.

Minesh Mathew, Dimosthenis Karatzas, and C. V. Jawahar. 2021b. Docvqa: A dataset for vqa on document images. *Preprint*, arXiv:2007.00398.

Jamshed Memon, Maira Sami, Rizwan Ahmed Khan, and Mueen Uddin. 2020. Handwritten optical character recognition (ocr): A comprehensive systematic literature review (slr). *IEEE access*, 8:142642–142668.

Anand Mishra, Shashank Shekhar, Ajeet Kumar Singh, and Anirban Chakraborty. 2019. Ocr-vqa: Visual question answering by reading text in images. In *ICDAR*.

Wenjun Peng, Guiyang Li, Yue Jiang, Zilong Wang, Dan Ou, Xiaoyi Zeng, Derong Xu, Tong Xu, and Enhong Chen. 2024. Large language model based long-tail query rewriting in taobao search. In *Companion Proceedings of the ACM Web Conference 2024*, pages 20–28.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652.

Yusuke Shinyama, Pieter Marsman, and pdfminer.six Contributors. 2019. The pdfminer.six library. Version 2019.

Manan Suri, Puneet Mathur, Franck Dernoncourt, Kanika Goswami, Ryan A Rossi, and Dinesh Manocha. 2024. Visdom: Multi-document qa with visually rich elements using multimodal retrieval-augmented generation. *arXiv preprint arXiv:2412.10704*.

Ryota Tanaka, Taichi Iki, Taku Hasegawa, Kyosuke Nishida, Kuniko Saito, and Jun Suzuki. 2025. Vdocrag: Retrieval-augmented generation over visually-rich documents. *arXiv preprint arXiv:2504.09795*.

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

Rubèn Tito, Dimosthenis Karatzas, and Ernest Valveny. 2023. Hierarchical multimodal transformers for multi-page docvqa. *Preprint*, arXiv:2212.05935.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022a. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. *arXiv preprint arXiv:2212.10509*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022b. Musique: Multihop questions via single-hop question composition. *arXiv preprint arXiv:2108.00573*.

Jordy Van Landeghem, Rubèn Tito, Łukasz Borchmann, Michał Pietruszka, Pawel Joziak, Rafal Powalski, Dawid Jurkiewicz, Mickaël Coustaty, Bertrand Anckaert, Ernest Valveny, and 1 others. 2023. Document understanding dataset and evaluation (dude). In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19528–19540.

Prakhar Verma, Sukruta Prakash Midigeshi, Gaurav Sinha, Arno Solin, Nagarajan Natarajan, and Amit Sharma. 2025. Plan*rag: Efficient test-time planning for retrieval augmented generation. *Preprint*, arXiv:2410.20753.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. Multilingual E5 text embeddings: A technical report. Technical report, Microsoft Research.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. 2023. Autogen: Enabling next-gen llm applications via multi-agent conversation. *Preprint*, arXiv:2308.08155.

Yiran Wu, Tianwei Yue, Shaokun Zhang, Chi Wang, and Qingyun Wu. 2024. Stateflow: Enhancing llm task-solving through state-driven workflows. In *First Conference on Language Modeling*.

Xudong Xie, Hao Yan, Liang Yin, Yang Liu, Jing Ding, Minghui Liao, Yuliang Liu, Wei Chen, and Xiang Bai. 2024. Wukong: A large multimodal model for efficient long pdf reading with end-to-end sparse sampling. *arXiv preprint arXiv:2410.05970*.

Xudong Xie, Hao Yan, Liang Yin, Yang Liu, Jing Ding, Minghui Liao, Yuliang Liu, Wei Chen, and Xiang Bai. 2025. Pdf-wukong: A large multimodal model for efficient long pdf reading with end-to-end sparse sampling. *Preprint*, arXiv:2410.05970.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

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

Wenhao Yu, Hongming Zhang, Xiaoman Pan, Kaixin Ma, Hongwei Wang, and Dong Yu. 2024b. Chain-of-note: Enhancing robustness in retrieval-augmented language models. *Preprint*, arXiv:2311.09210.

Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. 2025. Deepresearcher: Scaling deep research via reinforcement learning in real-world environments. *Preprint*, arXiv:2504.03160.

A Appendix

Table 7: All-Match Retrieve Rate on MMLongBench with two retrieve models. A question is all-match if all ground-truth evident pages is present in the retrieved pages. Note that ColQwen-2.5 (v0.2) is trained with strategy introduce by ColPali.

Model	Top-K	Match Rate %
ColQwen-2.5	2	54.55
ColPali	2	28.74
ColQwen-2.5	6	70.13
ColPali	6	44.35
ColQwen-2.5	10	79.22
ColPali	10	55.15

A.1 Pilot Study

We perform a pilot experiment on MMLongBench to understand how VLM performs on DocVQA problems. To compare, we test `Qwen2.5-VL-32B` with no evidence page and with ground-truth evidence pages provided by the dataset. To understand how different modalities of evidences affect the results, we also input image of the pages, text of the pages (extracted with PDF tools), and both text and image of the pages. We find that using the image form of ground-truth pages is crucial, since there is 25% accuracy gap between image-based and text-based input. Combining the two forms can further boost the performance, but are not significant.

Table 8: Model accuracies by input type (values to be filled)

Doc Type	Model	Accuracy (%)
N/A	Qwen2.5-VL-32B	22.18
GT Image	Qwen2.5-VL-32B	67.94
GT Text	Qwen2.5-VL-32B	42.40
GT Both	Qwen2.5-VL-32B	69.06

A.2 Usage of AI assistant

We use AI assistant to help debug code and build utility functions. We also use AI assistant to refine writing.

A.3 Detailed Retrieval Metric Calculation

Let $\mathcal{Q}$ be the set of N evaluation questions. For every question $q \in \mathcal{Q}$ we denote by

$$G_q \subseteq \mathcal{D}, \qquad R_q \subseteq \mathcal{D}$$

the *gold* set of truly relevant pages and the *retrieved* set (the top–k pages produced by the system).

All-hit Rate (Coverage) The *all-hit rate* measures the proportion of questions for which *every* gold page is retrieved:

$$\text{AllHit} \;=\; \frac{\big|\{\, q \in \mathcal{Q} \;:\; G_q \subseteq R_q \,\}\big|}{N}.$$

Because a single missing page makes a query count as a failure, All Hit captures strict evidence coverage.

Page-level F_1 (Retrieval Efficiency) Retrieval may also be viewed as a binary decision for each candidate page (gold vs. non-gold). For every question we compute *precision* and *recall*, abbreviated P_q and R_q:

$$P_q \;=\; \frac{|G_q \cap R_q|}{|R_q|}, \qquad R_q \;=\; \frac{|G_q \cap R_q|}{|G_q|}.$$

Their harmonic mean gives the question-level F_1:

$$\text{F1}_q \;=\; \begin{cases} \dfrac{2\,P_q\,R_q}{P_q + R_q}, & \text{if } P_q + R_q > 0, \\ 0, & \text{otherwise.} \end{cases}$$

Macro-averaging over questions yields the final score:

$$\text{PageF1} \;=\; \frac{1}{N} \sum_{q \in \mathcal{Q}} \text{F1}_q.$$

A.4 Additional Analysis

Role of the Memory Module We evaluated the effect of the memory accumulation component by disabling it during iterative reasoning.

Setting	QA Accuracy
SimpleDoc (with memory)	59.55
SimpleDoc (without memory)	59.27

Table 9: Effect of the memory module on MMLongBench.

The absence of memory results in a small accuracy decrease, as more than half of the tasks are solved in a single iteration. However, the memory module becomes critical in multi-step questions, allowing the agent to maintain evidence across iterations and guide subsequent retrievals.

Computational Statistics and Pipeline Simplicity We report token-level I/O for both SimpleDoc and a multi-agent baseline (MDocAgent) to compare computational cost.

- For VLMs: input token length = sum of retrieved image-text pages + query + memory.
- For text-only LLMs: summary-level retrieval + re-ranking only.

We disregard one-time operations such as document preprocessing and query embedding, as they are amortized over the dataset and use small models. These statistics are compiled under a fixed dataset (MMLongBench) for both SimpleDoc and MDocAgent.

Model	Qwen2.5-VL-32B		Qwen3-30B	
	Input	Output	Input	Output
SimpleDoc	4542.67	301.12	3164.83	187.43
MDocAgent	6301.48	346.44	4581.52	211.56

Table 10: Token statistics on MMLongBench (average per question).

Despite using fewer pages on average (3.5 per question), SimpleDoc achieves higher accuracy while maintaining a streamlined, single-agent pipeline.

A.5 Error Analysis

To better understand system behavior, we categorized observed failures into the following types and provide representative examples for each.

Retrieval Failure Relevant pages were not retrieved.
doc_id: fdac8d1e9ef56519371df7e6532df27d.pdf
Question: How many organisations does this report introduce in detail (at least one paragraph)?
Original Answer: 19
Final Answer: The report introduces 8 organizations in detail (ISRO, DOS, VSSC, NARL, NE-SAC, Antrix, SCL, IIST).
Gold Pages: [9, 17, 18, 19, 20, 21]
Retrieved Pages: [9, 17, 18, 19, 20]

Partial Evidence Retrieval Only a subset of gold evidence pages was retrieved.
doc_id: Independents-Report.pdf
Question: What's the percentage of people who are Democrats and voted in the last election compared to the entire population in 2018?
Original Answer: 18.29%
Final Answer: The document provides turnout rates for Democrats (59%) and Democratic-leaning independents (48%) but does not explicitly state the total population's voting rate.
Gold Pages: [4, 5, 6]
Retrieved Pages: [4, 5]

Hallucination Answers not grounded in retrieved content.
doc_id: ddoseattle-150627210357-lva1-app6891_95.pdf
Question: Which YouTube video do the slides use to show the consequence of having a weak data leadership?
Original Answer: Not answerable
Final Answer: Referenced the YouTube video "Girls Crash into Lake following Bad GPS directions" (Page 50).
Gold Pages: [-]
Retrieved Pages: [50]

Long-Context Overload Important content diluted due to long input sequences.
doc_id: bdf54dxa.pdf
Question: How many possible problems does the dishwasher encounter before calling for service?
Original Answer: 17
Final Answer: Retrieved pages contained empty tables; no specific problems or error codes were visible.
Gold Pages: [29, 30, 31, 51]
Retrieved Pages: [29, 30, 31, 51]

Multi-modal Misalignment Conflicting interpretations between visual and textual cues (e.g., chart vs. text).
doc_id: PH_2016.06.08_Economy-Final.pdf
Question: Among all charts in this report, how many compare the general public with Latinos/Hispanics?
Original Answer: 6
Final Answer: Visual analysis of pages 4, 6, 7, 9, 10, 13, 14 yields 7 charts matching the criteria.
Gold Pages: [4, 6, 7, 9, 10, 13, 14]
Retrieved Pages: [4, 6, 7, 9, 10, 13, 14]

Ambiguous Question Interpretation Multiple plausible interpretations of a particular question.
doc_id: PH_2016.06.08_Economy-Final.pdf
Question: Among the Hispanics shaped by four current financial situation, which situation has the highest percentage expecting their future financial situation to get a lot worse?
Original Answer: Poor financial condition
Final Answer: 8% of both "Only fair" and "Poor" categories expect their situation to get a lot worse.
Gold Pages: [16]
Retrieved Pages: [16]

Unanswerable Question Questions where no answer exists in the document.
doc_id: edb88a99670417f64a6b719646aed326.pdf
Question: Is there blue color handwritten words in page 30?
Original Answer: Not answerable
Final Answer: The document contains only pages 1–20 and no references to blue handwritten words; page 30 does not exist.
Gold Pages: [–]
Retrieved Pages: [–]

Layout Error Misinterpretation of table structures or complex layouts.
doc_id: finalmediafindingspdf-141228031149-conversion-gate02_95.pdf
Question: How many female respondents in Wave III never listen to the radio in the last half-year?
Original Answer: 1115
Final Answer: The document provides 55.7% but not the total number of female respondents, making the absolute count unanswerable.
Gold Pages: [16]
Retrieved Pages: [16]

A.6 Prompts Used in SimpleDoc

In Figure 4, we show the prompt for pre-processing each page. In Figure 5, we show the prompt to retrieve pages based on reasoning. In Figure 6, we should the prompt for the reasoner agent.

Page Index Prompt:
You are tasked with creating a comprehensive summary of a given page from a document. Your summary should focus on extracting and describing the main content, tables, figures, and images present on the page.

Raw text extracted from the retrieved pages (without visual information):
<page_text>
{PAGE_TEXT}
</page_text>
Please follow these steps to create your summary:
1. Carefully read and analyze the page content.
2. Identify the main topics, key points, and important details presented on the page.
3. Note any tables, figures, charts, diagrams, or images on the page and briefly describe their content and purpose.
4. Create a structured summary that captures:
- The essential textual information from the page
- Descriptions of any visual elements (tables, figures, images, etc.)
- Any particularly notable or unique information

Present your summary within <summary> tags. The summary should be concise yet comprehensive, typically 5-8 sentences for text-only pages, with additional sentences as needed to describe visual elements.
For visual elements, please use these specific tags:
- <table_summary> for descriptions of tables
- <figure_summary> for descriptions of figures, charts, graphs, or diagrams
- <image_summary> for descriptions of photos, illustrations, or other images

Example structure:
<summary> [Main text content summary here]

<table_summary> Table 1: [Brief description of what the table shows] </table_summary>
<figure_summary> Figure 2: [Brief description of what the figure depicts] </figure_summary>
<image_summary> [Brief description of image content] </image_summary> </summary>

Figure 4: Page indexing prompt used to extract structured information from document pages.

Page Retrieval Prompt:
You are a document understanding agent tasked with identifying the most promising page(s) for a given user query. You will be presented with summaries of each page in a document and a user query. Your task is to determine which page(s) should be examined in detail in a subsequent step.

First, review the summaries of each page in the document:

<page_summaries> PAGE_SUMMARIES </page_summaries>
Now, consider the following user query:

<user_query>
USER_QUERY
</user_query>
Important context about your task:
1. You are performing an initial screening of pages based on limited information (summaries only).
2. The pages you select will be analyzed in depth by another agent who will have access to the full page content.
3. These summaries are inherently incomplete and may miss details that could be relevant to the query.
4. It's better to include a potentially relevant page than to exclude it at this stage.

To determine which pages warrant closer examination:

1. Identify keywords, topics, and themes in the query that might appear in the document.
2. Select any page(s) whose summaries suggest they might contain information related to the query.
3. Be inclusive rather than exclusive - if a page seems even somewhat related or contains terminology connected to the query, include it for further analysis.
4. Always select at least one page, even if the connection seems tenuous - the detailed examination will determine true relevance.
5. The page order should be from most relevant to less relevant in your answer.

Additionally, create a comprehensive document-level summary that addresses the user query based on your understanding of the entire document. This summary should:
1. Provide a high-level perspective on how the document relates to the query
2. Synthesize relevant information across multiple pages
3. Highlight key concepts, definitions, or facts from the document that pertain to the query
4. Outline a strategic approach to solving the query based on the document's content
5. Identify potential solution paths and the types of information that should be prioritized
6. Do not be too certain about the conclusions drawn from the summaries, as they may not capture all relevant details
7. Be concise but informative (5-8 sentences)

After your analysis, provide your final answer in the following format:

<document_summary> [A comprehensive summary addressing how the document relates to the user query...] </document_summary>
<selected_pages> [List the indices of selected pages, separated by commas if there are multiple] </selected_pages>

Figure 5: Prompt for selecting top pages to retrieve for downstream reasoning.

Question Answering Prompt:
You are an AI assistant capable of analyzing documents and extracting relevant information to answer questions. You will be provided with document pages and a question about these pages.

Consider this question about the document:
<question> QUESTION </question>
Document level summary:
<document_summary> DOCUMENT_SUMMARY /document_summary>
The page numbers of the CURRENT RETRIEVED PAGES that you should analyze:
<retrieved_pages> RETRIEVED_PAGE_NUMBERS </retrieved_pages>
Raw text extracted from the retrieved pages (without visual information): <page_text> PAGE_TEXT </page_text>
IMPORTANT: Images of the retrieved pages are attached at the end of this prompt. The raw text extracted from these images is provided in the <page_text> tag above. You must analyze BOTH the visual images AND the extracted text, along with the <document_summary>, to fully understand the document and answer the question accurately.

<scratchpad> 1. List key elements from text and images
2. Identify specific details that relate to the question
3. Make connections between the document information (from both images, text, summary) and the question 4. Determine if the provided information is sufficient to answer the question 5. If you believe other pages might contain the answer, be specific about which content you're looking for that hasn't already been retrieved </scratchpad>
CRITICAL INSTRUCTION: First carefully check if:

The pages listed in <retrieved_pages> are already the specific pages that would contain the answer to the question
The specific tables, figures, charts, or other elements referenced in the question are already visible in the current images
The document summary explicitly mentions the content you're looking for
Do not request these same pages or elements again in a query update.

Based on your analysis in the scratchpad, respond in one of three ways:

If the provided pages contain sufficient information to answer the question, or if the document summary clearly indicates the answer to the question is that something does not exist:
<answer> Your clear and concise response that directly addresses the question, including an explanation of how you arrived at this conclusion using information from the document. </answer>
If based on the document summary and current pages, you're confident the entire document likely doesn't contain the answer, OR if the specific pages/tables/figures/elements that should contain the answer are already in the current context but don't actually contain relevant information:
<not_answerable> The document does not contain the information needed to answer this question. </not_answerable>
If based on the document summary, you believe the answer exists in other parts of the document that haven't been retrieved yet:
<query_update> [Provide a rewritten long query that PRESERVES THE ORIGINAL MEANING of the question but adds specific details or keywords to help retrieve new relevant pages. The information retrieved from this new query must directly answer the original question.] </query_update>
<notes> [IF using query_update, provide concise notes about what you've learned so far, what information is still missing, and your reasoning for the updated query. These notes will be appended to the document summary in the next iteration to maintain context across searches.] </notes>
Usage guidelines:

Use <answer> when you can answer the question with the provided pages, OR when you can determine from the document summary that the answer is that something doesn't exist.

Use <not_answerable> when either: The document summary and current pages together suggest the document as a whole doesn't contain the answer
OR the specific pages that should logically contain the answer are already provided in <retrieved_pages> but don't actually have the relevant information

OR specific tables, figures, charts, or elements mentioned in the question are visible in the current pages but don't contain the information being asked for
Use <query_update> ONLY when seeking information you believe exists in other pages that have NOT already been retrieved. Never request pages that are already listed in <retrieved_pages> or elements already visible in the current context. When creating a <query_update>, you MUST preserve the original meaning and intent of the question while adding specific details, keywords, or alternative phrasings that might help retrieve the necessary information. The answer to your new query must directly answer the original question. When using <query_update>, ALWAYS include the <notes> tag to summarize what you've learned so far and explain your reasoning for the updated query.
Your response must include both the <scratchpad> tag and exactly one of the following tags: <answer>, <not_answerable>, or <query_update>. If you use <query_update>, you must also include the <notes> tag.
<answer> / <not_answerable> / <query_update>

Figure 6: Prompt used during the question-answering stage, leveraging both extracted text and page images.