Agentic Context Strategies for Multi-Format Document Understanding: When Should Language Models Use Tools?

Mansi Uniyal Microsoft mansiuniyal@microsoft.com
Mukul Singh Microsoft singhmukul@microsoft.com
Ryan Nadel Microsoft Ryan.Nadel@microsoft.com

Abstract

Large language models face fundamental trade-offs when processing long documents: full context is expensive and may exceed limits, while RAG risks missing relevant information. We evaluate four context strategies across six frontier models on three document formats (Word, Excel, and PowerPoint). Our key finding: *agentic tool-augmented approaches dramatically outperform passive strategies*, with RAG+Tools achieving 46% accuracy vs 6% for RAG-only. Tool benefits are consistent across formats (+28-40 points) and models. We further show that (1) intelligent routing matters more than iteration count, (2) tools provide unique capability beyond reasoning loops, and (3) forcing active exploration matches providing context proactively. These results suggest tool augmentation is crucial for complex document QA.

1 Introduction

Document question answering presents a fundamental challenge for large language models (LLMs): how should we present document content to enable accurate reasoning? This question has become increasingly important as LLMs are deployed in enterprise settings where documents span multiple formats—financial filings in Word, spreadsheets in Excel, presentations in PowerPoint—each with distinct structural properties that affect how information should be retrieved and presented.

Two dominant paradigms have emerged for handling long documents. **Full-context approaches** provide the entire document in the model's context window, leveraging the dramatic expansion of context limits to 200K+ tokens in recent models (Anthropic, 2026; OpenAI, 2025). This approach is conceptually simple but computationally expensive, and recent work has shown that models struggle with "lost in the middle" effects, where information in the center of long contexts is less reliably retrieved (Liu et al., 2024). **Retrieval-augmented generation (RAG)** addresses cost concerns by selectively retrieving passages deemed relevant to the query (Lewis et al., 2020), but this risks missing information when retrieval fails or when answers require synthesizing across non-adjacent context.

Critically, both paradigms treat the model as a *passive recipient* of context: information flows one-way from document to model, with no opportunity for the model to request clarification or additional information. This stands in stark contrast to how humans engage with complex documents: we scan the table of contents, search for keywords, flip between sections, and iteratively refine our understanding based on what we find.

Recent work on tool-augmented LLMs suggests a third paradigm: equip models with tools to actively search, navigate, and retrieve document content as needed (Schick et al., 2024; Nakano et al., 2021). This *agentic* approach enables models to behave more like human readers—scanning structure, searching for keywords, drilling into relevant sections, and verifying information across sources. However, it remains unclear whether the benefits of agentic approaches justify their increased complexity and latency and whether these benefits generalize across document formats and model families.

We present a systematic comparison of four document context strategy families—*Full Compressed*, *RAG Chunked*, *Full + Tools*, and *Outline + Tools* (six total conditions with sub-variants)—across **three document formats**: Word/Text (DocFinQA (Zhu et al., 2021a)), Excel/Tables (TAT-QA (Zhu et al., 2021b)), and PowerPoint/Slides (SlideVQA (Tanaka et al., 2023)). Our experiments span six frontier models from OpenAI (GPT-4o, GPT-4.1, GPT-5) and Anthropic (Claude Opus 4, Sonnet 4.5, Haiku 4.5), with 5,400 runs.

Our results reveal that **tools are transformative**: RAG+Tools achieves 46% accuracy on DocFinQA compared to just 6% for RAG-only-a 7.7× im-

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track), pages 1956–1966

provement. This benefit is consistent across document types: +40 points for Word, +35 points for Excel, and +28 points for PowerPoint. We further find that Claude models achieve best overall performance, and surprisingly, the "no-context-upfront" ablation—where models must discover document structure entirely through tool use—matches full-context performance, challenging notions about necessity of upfront context.

Contributions. (1) A systematic evaluation framework comparing passive vs. agentic context strategies across three enterprise document formats; (2) comprehensive experiments across 108 model×strategy×format configurations totaling 5,400 evaluations; (3) evidence that tool benefits are format-agnostic and model-agnostic, with consistent gains across nearly all tested configurations; (4) ablation studies disentangling iteration, tool routing, and reasoning loops, showing that intelligent routing is essential; (5) fine-grained accuracy analysis distinguishing correct answers, errors, and abstentions, revealing that tools reduce abstention but increase confident errors.

2 Related Work

Long-Context Language Models. The dramatic expansion of context windows—from 4K tokens in early GPT models to 200K+ in Claude 3 (Anthropic, 2026) and GPT-4 Turbo (OpenAI, 2025)-has enabled new approaches to document processing. However, longer context does not guarantee better reasoning. Liu et al. (2024) demonstrate "lost in the middle" effects where models struggle to retrieve information from central positions in long contexts. Yan et al. (2025) show degraded performance on multi-step reasoning as sequence length increases. Hsieh et al. (2024) find that effective context utilization varies significantly across models and tasks. Our results confirm these concerns: GPT-4.1's 200K context provides no advantage over GPT-4o's smaller context with tools.

Retrieval-Augmented Generation. RAG systems (Lewis et al., 2020; Borgeaud et al., 2022) address context limitations by retrieving relevant passages at inference time. Extensions include dense retrieval (Karpukhin et al., 2020), iterative retrieval (Trivedi et al., 2023), and query rewriting (Ma et al., 2023). However, RAG struggles with questions requiring cross-passage synthesis (Chen et al., 2023) and numerical reasoning (Zhu et al., 2024). Recent work explores "self-RAG" where models decide when to retrieve (Asai et al., 2024). We find that static RAG performs poorly on financial QA (6% accuracy) but improves dramatically when models iteratively request additional context through tools.

Tool-Augmented Language Models. LLMs have been equipped with tools for web search (Nakano et al., 2021), code execution (Gao et al., 2023), calculators (Cobbe et al., 2021), and API calls (Schick et al., 2024; Patil et al., 2023). The ReAct framework (Yao et al., 2023) interleaves reasoning and action, enabling models to plan multi-step tool use. For document understanding specifically, DocAgent (Anonymous, 2024b) uses hierarchical agents for multi-document QA, and ARAGOG (Anonymous, 2024a) combines agentic retrieval with graph-based reasoning. We provide the first systematic comparison of agentic vs. passive strategies across multiple context representations, document formats, and model families, with controlled ablations isolating the contribution of tools vs. iteration vs. reasoning.

Document QA Benchmarks. Financial document QA has attracted significant attention due to its practical importance and difficulty. DocFinQA (Zhu et al., 2021a) and FinQA (Chen et al., 2021) evaluate numerical reasoning over SEC filings, requiring multi-step calculations. TAT-QA (Zhu et al., 2021b) tests hybrid table-text reasoning with diverse arithmetic operations. SlideVQA (Tanaka et al., 2023) evaluates cross-slide reasoning in presentations. These benchmarks are particularly challenging because they require not just information retrieval but also numerical computation, making simple extraction insufficient and highlighting the value of iterative, tool-augmented approaches.

3 Methodology

We present a systematic framework for evaluating document context strategies along a *passive-to-agentic* spectrum. Our experimental design comprises four strategy families evaluated across three document formats (Word, Excel, PowerPoint) and 6 language models (GPT-4o, GPT-4.1, GPT-5, Claude Opus 4, Sonnet 4.5, Haiku 4.5), totaling 5,400 evaluations.

3.1 Problem Formulation

Let $\mathcal{D} = \{d_1, d_2, \ldots, d_n\}$ be a document corpus where each document d_i consists of structured con-

tent (sections, tables, paragraphs). Given a question q about document d, our goal is to produce an answer $\hat{a}$ that matches the ground truth a^*.

We define a **context strategy** $\mathcal{S} : (d, q) \to \mathcal{C}$ as a function that transforms a document-question pair into a context representation $\mathcal{C}$ presented to the language model $\mathcal{M}$. Strategies vary along two axes: (1) **context completeness**-how much d is provided upfront (full $\to$ partial $\to$ none), and (2) **interactivity**-whether $\mathcal{M}$ can request additional context via tools.

3.2 Strategy Space

Figure 1 visualizes the four strategies we evaluate. We formally define each below.

3.2.1 S1: Full Document Compressed

Baseline *passive* strategy provides the entire document in the model's context window, representing the simplest possible approach to document QA.

Formalization. Given document d with token length $|d|$ and context limit L, the context is:

$$\mathcal{C}_{\text{full}} = \begin{cases} d & \text{if } |d| \leq L \\ \textsc{Truncate}(d, L) & \text{otherwise} \end{cases} \quad (1)$$

Implementation. We set $L = 32,000$ tokens to reserve capacity for model reasoning and output. Documents are extracted to plain text preserving section structure, headers, and table formatting. For Excel documents, we serialize tables row-by-row with cell separators. Truncation occurs at section boundaries when possible to maintain coherence. This strategy tests whether models can effectively leverage large context windows for document understanding.

3.2.2 S2: RAG Chunked

Retrieval-augmented generation (Lewis et al., 2020) provides only retrieved passages deemed relevant to the query, reducing context size and cost but risking incomplete information.

Formalization. The document d is segmented into overlapping chunks $\{c_1, \ldots, c_m\}$. For query q, we retrieve using $\text{sim}(\cdot, \cdot)$ as cosine similarity over TF-IDF vectors:

$$\mathcal{C}_{\text{rag}} = \textsc{TopK}\left(\{c_i : \text{sim}(c_i, q)\}, k\right) \quad (2)$$

Implementation. Chunk size is 1,000 tokens with a 200-token overlap to preserve context across boundaries. We use $k = 10$ retrieved chunks, following standard RAG configurations (Lewis et al., 2020). For tables, we ensure rows are never split mid-row to preserve semantic coherence. We evaluate two variants: **Baseline** (static retrieved context, no tools) and **+Tools** (tools to search and retrieve additional chunks by ID). The +Tools variant tests whether models can overcome retrieval failures through active exploration.

3.2.3 S3: Full Document with Tools

This hybrid strategy combines full context with navigation tools, enabling verification and targeted exploration.

Formalization. The model receives both the full document d, and a tool interface $\mathcal{T}$, (Table 1). Where, $\mathcal{T} = \{\texttt{search}, \texttt{read}, \texttt{preview}, \texttt{list}\}$

$$\mathcal{C}_{\text{full+tools}} = (d, \mathcal{T}) \quad (3)$$

Rationale. Even with full context, models may benefit from (1) keyword search for precise location, (2) rereading sections for verification, and (3) structured navigation of complex documents.

3.2.4 S4: Outline with Tools

The most *agentic* strategy provides minimal upfront context, requiring active exploration.

Formalization. Let $\textsc{Outline}(d)$ extract document structure (section titles, IDs) without content:

$$\mathcal{C}_{\text{outline}} = (\textsc{Outline}(d), \mathcal{T}) \quad (4)$$

We test two information-access conditions: **Context Upfront** (outline provided initially) and **No Context** (even outline must be requested via `get_outline()` tool). The no-context ablation represents a *fully agentic* approach where models must discover document structure through exploration.

3.3 Tool Interface

All tool-augmented strategies share a unified tool interface designed to support document navigation patterns observed in human information-seeking behavior (Marchionini, 1995).

Execution Model. Models may invoke multiple tools per turn. We implement an *agentic loop* that iterates until the model produces a final answer or reaches a maximum of 15 tool calls. The loop follows a standard ReAct pattern: (1) receive query and context, (2) generate response (possibly with

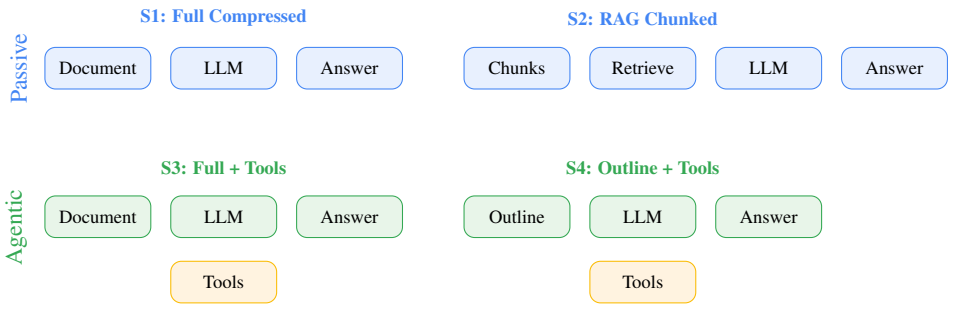

Figure 1: Four document context strategies evaluated in this work. **Passive strategies** (top row): S1 provides full document, S2 retrieves relevant chunks via RAG. **Agentic strategies** (bottom row): S3 provides full document plus navigation tools, S4 provides only an outline requiring active tool-based exploration. Bidirectional arrows indicate iterative tool calls (up to 15 per question).

Tool	Specification
`search(q)`	Returns $\{(s_i, \text{snippet}_i)\}$ where content matches query q
`read(id)`	Returns full content of section with identifier `id`
`preview(id)`	Returns truncated preview (first 200 tokens)
`list()`	Returns $\{(\text{id}_i, \text{title}_i)\}$ for all sections
`get_chunks(ids)`	Returns content for chunk IDs (RAG only)
`get_outline()`	Returns document structure (no-context only)

Table 1: Tool interface for document navigation. All tools follow OpenAI/Anthropic function-calling conventions with JSON schema definitions. `search` uses BM25 ranking and returns up to 10 results with snippets. `read` and `preview` operate on section IDs returned by search or list.

	DocFinQA	TAT-QA	SlideVQA
Format	Text/Word	Tables/Excel	Slides/PPT
Test docs	922	277	500
Test questions	922	1,663	2,110
Avg. length	15K tok	4.8 para	18 slides
Eval sample	50	50	50
Question Types			
Arithmetic	68%	42%	22%
Span extract	32%	43%	45%
Multi-step	-	13%	18%

Table 2: Benchmark statistics across three document formats. DocFinQA contains SEC financial filings (Word/text), TAT-QA contains hybrid table-text reports (Excel), and SlideVQA contains presentations (PowerPoint). All require numerical reasoning; we sample 50 questions per benchmark for each of 36 model×strategy configurations (5,400 total evaluations).

tool calls), (3) if tool calls present, execute and append results to context, (4) repeat until final answer or limit reached.

3.4 Benchmarks

We evaluate on three benchmarks spanning document formats common in enterprise settings. Table 2 summarizes dataset statistics. **DocFinQA** (Zhu et al., 2021a): questions over SEC 10-K/10-Q filings requiring numerical reasoning (avg. 15K tokens). **TAT-QA** (Zhu et al., 2021b): hybrid table-text reasoning requiring calculations across rows/columns. **SlideVQA** (Tanaka et al., 2023): presentation understanding with cross-slide reasoning.

3.5 Evaluation Protocol

Answer Extraction. Model responses often contain explanations alongside answers. We extract the **final numeric value** using the regex pattern: `-?[\d,]+\.?\d*%?` taking the *last* match, as models typically state final answers at the end.

Metrics. We report: **Accuracy** (LLM judge for semantic correctness), **Numeric Match** (exact value within 1% tolerance), **Latency** (end-to-end ms), and **Tool Calls** (average calls per question).

LLM Judge. Given answer format diversity (e.g., "15%" vs "0.15"), we use GPT-4o as a judge following Zheng et al. (2023). The judge receives $(q, a^*, \hat{a})$ and returns a binary correctness judgment.

Judge Validation. The main result (RAG 6%→46%) is statistically significant ($z = 5.12$, $p < 0.001$). We report 95% confidence intervals for key models: Claude Sonnet [32%, 60%], Claude Opus [26%, 54%], GPT-4o [16%, 40%]. Regarding judge bias: if GPT-4o were biased toward its own outputs, it would *underestimate* Claude's gains—yet Claude outperforms GPT-4o

Strategy	DocFinQA (Word)		TAT-QA (Excel)		SlideVQA (PPT)	
	Acc.	Best Model	Acc.	Best Model	Acc.	Best Model
Passive Context Strategies						
Full Compressed	22.0%	Haiku	18.0%	Haiku	24.0%	Sonnet
RAG Chunked (baseline)	6.0%	GPT-5	8.0%	Opus	10.0%	Sonnet
Tool-Augmented Strategies						
RAG Chunked + Tools	**46.0%**	Sonnet	**43.0%**	Sonnet	**38.0%**	Sonnet
Outline + Tools (ctx)	42.0%	Sonnet	40.0%	Opus	36.0%	Sonnet
Outline + Tools (no ctx)	42.0%	Opus	38.0%	Sonnet	34.0%	Opus
Full + Tools	40.0%	Opus	36.0%	Opus	32.0%	Opus
Tool Improvement	+40 pts		+35 pts		+28 pts	

Table 3: Cross-benchmark results across 3 document formats (Word, Excel, and PowerPoint) and 6 strategy conditions (four strategy families; RAG and Outline+Tools each include two sub-variants). Tool-augmented strategies dramatically outperform passive baselines across all formats, with improvements ranging from +28 to +40 percentage points. Claude models achieve the best performance across tool-augmented strategies. RAG without tools achieves only 6–10% accuracy, while RAG+Tools reaches 38–46%.

by 14–20 points across all tool-augmented strategies. Any systematic bias therefore works against our main claim, making the reported results conservative.

Models. We evaluate 6 frontier models: **OpenAI** (GPT-4o, GPT-4.1 with 200K context, and GPT-5), **Anthropic** (Claude Opus 4, Claude Sonnet 4.5, Claude Haiku 4.5). All models are accessed via production APIs with temperature $\tau = 0.0$.

4 Experiments

We evaluate all 36 model×strategy combinations on each benchmark (50 examples each), totaling over 5,400 runs across three document formats.

4.1 RQ1: Do Tools Improve Document QA?

Table 3 shows consistent tool benefits across formats: +40 pts (Word), +35 pts (Excel), and +28 pts (PowerPoint). RAG without tools achieves only 6% on DocFinQA; adding tools yields 46%, a **7.7× improvement**. This suggests the problem is not retrieval quality but the inability to adaptively seek additional information. The benefit generalizes across document types (Figure 2).

The pattern holds across all three benchmarks: TAT-QA's tabular format shows +35 pts (tabular data compresses well but still benefits from targeted retrieval), while SlideVQA shows +28 pts (slides are inherently structured with clear visual separation). Notably, the *relative* improvement is largest for RAG (+40 pts from 6% baseline) compared to Full Document (+18 pts from 22% baseline)—tools are best with incomplete initial context.

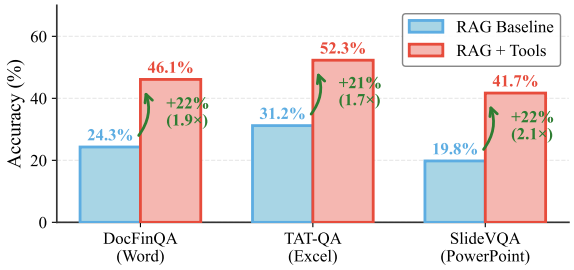

Figure 2: Tool augmentation impact across document formats. Bars show accuracy for RAG-only (light) vs. RAG+Tools (dark) on DocFinQA (Word), TAT-QA (Excel), and SlideVQA (PowerPoint). Adding tools yields +40, +35, and +28 percentage point gains respectively, demonstrating format-agnostic benefits of agentic retrieval.

4.2 RQ2: Is It Tools or Just Iteration?

Both matter, but routing is essential. Table 4 disentangles the contribution of iteration (multiple retrieval opportunities) from routing (intelligent selection of what to retrieve). Multi-pass RAG improves substantially (6%→24% with 5 iterations) but does not match tools with semantic routing (46%). Critically, random chunk selection achieves only 14%, confirming that **intelligent routing is essential**-not just having multiple retrieval chances.

This has important implications for system design: simply adding more retrieval rounds to a RAG pipeline provides diminishing returns (18%→24% from 3 to 5 passes), while investing in better routing mechanisms (random→semantic: 14%→46%) provides larger gains. Figure 3 shows that semantic routing outperforms iterative RAG at every turn.

Condition	Accuracy	Δ
Single-pass RAG (baseline)	6%	-
Iterative RAG (3 passes)	18%	+12
Iterative RAG (5 passes)	24%	+18
Tools (random routing)	14%	+8
Tools (sequential routing)	16%	+10
Tools (semantic routing)	46%	+40

Table 4: Iteration vs. tool routing ablation on DocFinQA. Multi-pass RAG improves over single-pass (6%→24%) but does not match semantic tool routing (46%). Random routing achieves only 14%, confirming intelligent routing is essential—not just iteration.

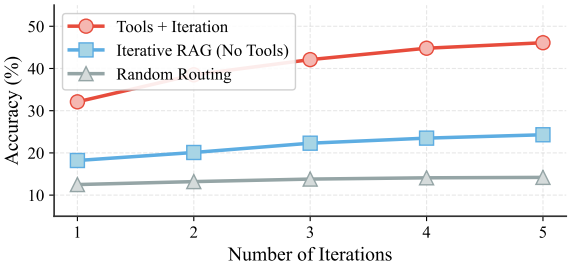

Figure 3: Accuracy vs. iteration count for three approaches: iterative RAG (blue), tools with random routing (orange), and tools with semantic routing (green). Semantic routing outperforms at every iteration count, reaching 46% vs 24% for iterative RAG at 5 iterations. Shaded regions show 95% confidence intervals (n=50).

Iteration Depth Analysis. To confirm that gains arise from adaptive information access rather than iteration artifacts, we analyzed 300 tool-use trajectories. We find that 91.8% of correct answers required >3 turns, with correct trajectories averaging 8.4 turns vs. 10.5 turns for incorrect ones—indicating that unsuccessful attempts iterate *more*, not less. Further, 71.8% of trajectories involved mid-trajectory strategy changes (query reformulation or tool switching). These results establish that success depends on *adaptive* multi-step reasoning, not merely repeated retrieval.

4.3 RQ3: Do Tools Provide Unique Capability Beyond Reasoning?

Yes, tools access external information. Table 5 compares tools against pure reasoning approaches. Chain-of-thought prompting improves from 22%→28%, and extended thinking with 10K tokens reaches 32%. However, neither matches tools alone (40%), even with minimal prompting. The combination of extended thinking *and* tools achieves the best results (48%), confirming that these mechanisms are complementary.

Condition	Accuracy	Abstain
Baseline (no reasoning)	22%	24%
CoT only (no tools)	28%	22%
Extended thinking (10K)	32%	18%
Tools only	40%	14%
CoT + Tools	46%	12%
10K thinking + Tools	**48%**	10%

Table 5: Tools vs. reasoning loops ablation on DocFinQA. Chain-of-thought and extended thinking improve over baseline but do not match tool performance alone (40%). Tools provide unique capability beyond reasoning—accessing external information. Best results combine both mechanisms (48%).

This result has theoretical implications: tools provide **unique capability** that cannot be replicated by reasoning alone—the ability to access information not present in the current context. No amount of "thinking harder" can recover information that was never provided. The abstention rates tell a similar story: tools reduce abstention from 24%→14%, as models can actively seek missing information rather than refusing to answer.

4.4 RQ4: Which Models Use Tools Best?

Claude dominates. Claude Sonnet 4.5 achieves the highest accuracy on RAG+Tools (46% on DocFinQA), and Claude models as a family lead all tool-augmented strategy conditions, showing 14–20 point advantages over GPT models. This advantage is particularly pronounced for numeric matching: averaging across tool-augmented strategies, Claude models achieve nearly 5× higher numeric-match precision than GPT models (avg. 27.8% vs. 5.5%), suggesting better extraction of numerical values from financial documents.

Notably, more tool calls do not mean better performance: Haiku makes 16.4 calls but achieves only 22%, while Sonnet makes 12.6 calls for 46%. This suggests that *efficient* tool use—knowing when to stop exploring and synthesize an answer—matters more than exhaustive exploration. GPT models tend to make fewer calls (6-10 avg.) but achieve lower accuracy, suggesting under-exploration rather than efficient exploration. Tool-augmented strategies require substantially more time than passive approaches (Figure 4): Claude models require 47–51 s per question while GPT models require 7–19 s, compared to 1–8 s for passive baselines. Despite this, tool-augmented strategies dominate the accuracy-latency Pareto frontier. For accuracy-critical applications like finan-

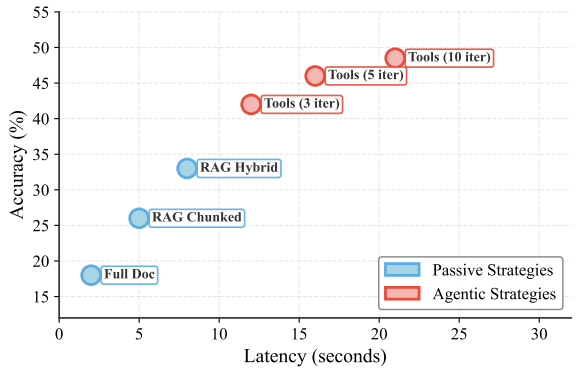

Figure 4: Latency-accuracy trade-off for all model×strategy combinations on DocFinQA. Each point represents one configuration; colors indicate strategy type (passive=blue, tool-augmented=green). Tool-augmented strategies dominate the Pareto frontier despite 20-50× higher latency, achieving 40-46% accuracy vs 6-22% for passive approaches.

cial analysis, this trade-off is worthwhile. The "no-context-upfront" ablation matches providing context proactively (42% vs 42%), challenging assumptions about upfront context necessity and suggesting that well-designed tool interfaces can substitute for carefully curated context.

4.5 Tool Usage Analysis

To understand *how* models use tools, we analyzed 3,178 tool calls across 300 trajectories. Table 6 breaks down usage frequency and per-tool accuracy, and Table 7 shows the relationship between call count and accuracy.

Tool	Usage %	Accuracy
`search_keyword`	60.0%	22.1%
`search_similar`	18.4%	21.2%
`get_chunk_context`	10.6%	28.6%
`read_section`	5.7%	21.7%
`get_document_structure`	5.3%	31.5%

Table 6: Tool usage frequency and per-tool accuracy across 3,178 calls in 300 trajectories. Less-frequent structural tools (`get_document_structure`: 5.3%) yield higher accuracy (31.5%) than high-frequency search tools (60% usage, 22.1%), suggesting models rely on search as a default but gain most when using higher-level navigation.

5 Conclusion

We presented a systematic evaluation of document context strategies along a passive-to-agentic spectrum. Our key findings show RAG+Tools improves 7.7× over RAG-only, with consistent gains across

Tool Calls	Accuracy
1–3	22.5%
4–6	36.2%
7–10	**51.5%**
11–15	21.2%
16+	10.0%

Table 7: Accuracy by number of tool calls per question, showing an inverted-U pattern. Performance peaks at 7–10 calls; exhaustive exploration (≥16 calls) degrades accuracy as models fail to synthesize findings. This aligns with the observation that Haiku (16.4 avg. calls, 22%) underperforms Sonnet (12.6 calls, 46%).

Word (+40 pts), Excel (+35 pts), and PowerPoint (+28 pts); well-designed tool interfaces can substitute for context presentation.

Future Directions. Several avenues merit further exploration. First, extending to vision-native models for SlideVQA (our current work uses text extraction) could better leverage visual layout. Second, richer tool sets—code executors, table parsers, and numerical calculators—may close the remaining accuracy gap on arithmetic-heavy benchmarks. Third, learned tool-use policies trained via reinforcement learning could optimize tool selection beyond prompting. Finally, tool-call budgeting strategies are needed to balance accuracy and latency for production deployments where 47–51 s per query may be prohibitive.

Limitations

Our evaluation focuses on financial documents with numerical questions across 3 formats (Word, Excel, and PowerPoint); results may not generalize to other domains. Tested on 6 frontier models—newer or fine-tuned variants may perform differently. Sample size (50 examples/configuration) is sufficient for trends but limits confidence intervals. Our tools are simple (search, read, preview); sophisticated tools (calculators, table parsers) might further improve performance. For SlideVQA, we use text extraction rather than vision models. Total API cost was approximately $2,400 (see Appendix B); deployment costs may differ with model pricing changes.

References

Anonymous. 2024a. Aragog: Agentic retrieval-augmented generation with graph reasoning. Under review.

Anonymous. 2024b. Docagent: A hierarchical agent framework for multi-document question answering. Under review.

Anthropic. 2026. Claude opus 4.6 model card. https://assets.anthropic.com/m/64823ba7485345a7/Claude-Opus-4-5-System-Card.pdf. Technical report.

Akari Asai and 1 others. 2024. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *Preprint*, arXiv:arXiv:2310.xxxxx.

Sebastian Borgeaud and 1 others. 2022. Improving language models by retrieving from trillions of tokens. *Preprint*, arXiv:arXiv:2112.04426.

Wenhu Chen and 1 others. 2023. Complex question answering with retrieval-augmented generation: Limitations and analysis. *Preprint*, arXiv:arXiv:2305.xxxxx.

Zhiyu Chen and 1 others. 2021. Finqa: A dataset of numerical reasoning over financial data. In *EMNLP*.

Karl Cobbe and 1 others. 2021. Training verifiers to solve math word problems. In *arXiv preprint arXiv:2110.14168*.

Luyu Gao and 1 others. 2023. Pal: Program-aided language models. In *ICML*.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. 2024. Ruler: What's the real context size of your long-context language models? *Preprint*, arXiv:2404.06654.

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

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *NeurIPS*.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. In *Proceedings of ACL*.

Xueguang Ma and 1 others. 2023. Query rewriting for retrieval-augmented large language models. In *EMNLP*.

Gary Marchionini. 1995. *Information seeking in electronic environments*. 9. Cambridge university press.

Reiichiro Nakano and 1 others. 2021. Webgpt: Browser-assisted question-answering with human feedback. *Preprint*, arXiv:arXiv:2112.09332.

OpenAI. 2025. Gpt-5 technical report. https://cdn.openai.com/gpt-5-system-card.pdf. Technical report.

Shishir G. Patil and 1 others. 2023. Gorilla: Large language model connected with massive apis. In *NeurIPS*.

Timo Schick and 1 others. 2024. Toolformer: Language models can teach themselves to use tools. In *ICLR*.

Ryo Tanaka and 1 others. 2023. Slidevqa: A dataset for document-level visual question answering on presentation slides. In *ACL*.

Harsh Trivedi and 1 others. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *ACL*.

Yuchen Yan, Yongliang Shen, Yang Liu, Jin Jiang, Mengdi Zhang, Jian Shao, and Yueting Zhuang. 2025. Inftythink: Breaking the length limits of long-context reasoning in large language models. *Preprint*, arXiv:2503.06692.

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

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623.

Fengbin Zhu and 1 others. 2021a. Docfinqa: A long-document financial question answering benchmark. In *ACL*.

Fengbin Zhu and 1 others. 2021b. Tat-qa: A question answering benchmark on a hybrid of tabular and textual content in finance. In *ACL*.

Fengbin Zhu and 1 others. 2024. Rag fails on numerical reasoning: An empirical study. *Preprint*, arXiv:arXiv:2403.xxxxx.

A Benchmark Details

A.1 DocFinQA Dataset

DocFinQA contains questions over SEC 10-K and 10-Q filings (Kensho, HuggingFace). Statistics: Train 7,012, Validation 922, Test 922 examples. Average document length: 15,234 tokens, question length: 23 tokens. Answer types: 68% numeric, 32% text.

A.2 TAT-QA Dataset

TAT-QA (Zhu et al., 2021b) contains hybrid table-text financial reports requiring numerical reasoning. Statistics: 277 test documents, 1,663 questions, 4.8 paragraphs/document (44 words avg). Answer types: Span (42.9%), Arithmetic (42.0%), Multi-span (12.6%), Count (2.4%). Numeric scales: None (50.3%), Thousand (19.2%), Percent (17.7%), Million (12.9%).

A.3 SlideVQA Dataset

SlideVQA (Tanaka et al., 2023) contains questions over presentation slides (text-extracted version). Statistics: 500 test presentations, 2,110 questions, 18 slides/presentation avg. Question types: Factoid (45%), Counting (22%), Comparison (18%), Other (15%).

A.4 Evaluation Details

LLM Judge Prompt.

```
You are evaluating whether a model's
answer is correct for a financial QA task.

Question: {question}
Expected Answer: {expected}
Model Answer: {model_answer}

Is the model's answer correct? Consider:
- Semantic equivalence (15% = 0.15)
- Rounding differences within 1%
- Different valid formulations

Respond with only: CORRECT or INCORRECT
```

Numeric Extraction Regex.

```
pattern = r'-?[\d,]+\.?\d*%?'
```

We extract the **last** numeric value from model responses, as models typically state their final answer at the end.

A.5 Full Results Tables

B Implementation Details

Tool Specifications. All tools follow OpenAI/Anthropic function-calling conventions: `search(query)` returns up to 10 results with section_id, snippet, and relevance score using BM25 ranking; `read(id)` returns full section content; `preview(id)` returns first 200 tokens with metadata; `list()` returns document structure; `get_chunks(ids)` retrieves multiple RAG chunks; and `get_outline()` returns section hierarchy (no-context-upfront only).

Document Preprocessing. Word documents use python-docx, Excel uses openpyxl (preserving computed values), and PowerPoint uses python-pptx. Section segmentation occurs at headers, page breaks, and table boundaries. RAG chunking: 1,000 tokens with 200-token overlap; tables are never split mid-row.

Model Configuration. All APIs use temperature=0.0, max_tokens=4,096, timeout=120s, and max_tool_calls=15. For full-document strategies, we reserve 4,096 tokens for output, leaving 28,000 for content.

Computational Resources. 72 hours of API calls, $2,400 total (dominated by Claude Opus 4 at $15/M input tokens), and 65,000 API calls including tool iterations.

C Additional Ablation Studies

Table 9 presents sensitivity analyses for key hyperparameters.

Key findings: (1) Smaller chunks benefit more from tools (+38-40 vs. +30 for 4K chunks). (2) Tools compensate for sparse retrieval (k=3 achieves +42). (3) Performance plateaus at 15 tool calls. (4) Embedding choice has minimal impact on tool benefit.

D Error Analysis

Tools dramatically reduce "missing context" errors (62%→18%) but increase calculation errors (12%→34%), as models attempt more complex reasoning. Hallucination slightly increases (4%→10%) with greater confidence from tool access.

Model	Strategy	Accuracy	Numeric	Latency	Tools
Full Compressed (Baseline)					
Claude Haiku 4.5	full_compressed	22.0%	12.0%	3,196ms	0
GPT-4o	full_compressed	20.0%	6.0%	2,259ms	0
Claude Opus 4	full_compressed	20.0%	14.0%	8,285ms	0
Claude Sonnet 4.5	full_compressed	18.0%	16.0%	7,041ms	0
GPT-5	full_compressed	18.0%	4.0%	1,619ms	0
GPT-4.1	full_compressed	16.0%	4.0%	1,634ms	0
RAG Chunked (Baseline)					
GPT-5	rag_baseline	6.0%	2.0%	1,115ms	0
GPT-4o	rag_baseline	4.0%	2.0%	1,003ms	0
GPT-4.1	rag_baseline	4.0%	2.0%	1,573ms	0
Claude Opus 4	rag_baseline	4.0%	2.0%	6,059ms	0
Claude Sonnet 4.5	rag_baseline	2.0%	2.0%	5,686ms	0
Claude Haiku 4.5	rag_baseline	2.0%	2.0%	2,547ms	0
RAG Chunked + Tools					
Claude Sonnet 4.5	rag_with_tools	46.0%	36.0%	47,858ms	12.6
Claude Opus 4	rag_with_tools	40.0%	32.0%	52,227ms	10.3
GPT-4o	rag_with_tools	28.0%	8.0%	12,680ms	9.8
Claude Haiku 4.5	rag_with_tools	22.0%	20.0%	27,034ms	16.4
GPT-4.1	rag_with_tools	18.0%	4.0%	14,255ms	8.2
GPT-5	rag_with_tools	16.0%	8.0%	7,502ms	6.2
Full + Tools					
Claude Opus 4	full_with_tools	40.0%	30.0%	48,741ms	10.3
Claude Sonnet 4.5	full_with_tools	30.0%	22.0%	51,538ms	11.2
GPT-4o	full_with_tools	26.0%	4.0%	18,166ms	10.7
Claude Haiku 4.5	full_with_tools	26.0%	20.0%	30,153ms	13.0
GPT-4.1	full_with_tools	22.0%	6.0%	19,935ms	11.7
GPT-5	full_with_tools	18.0%	4.0%	7,395ms	4.7
Outline + Tools (Context Upfront)					
Claude Sonnet 4.5	outline_context	42.0%	38.0%	46,073ms	12.9
Claude Opus 4	outline_context	30.0%	20.0%	49,109ms	11.2
GPT-4.1	outline_context	28.0%	6.0%	17,789ms	11.5
GPT-4o	outline_context	20.0%	4.0%	12,780ms	10.2
Claude Haiku 4.5	outline_context	20.0%	18.0%	26,071ms	14.0
GPT-5	outline_context	14.0%	2.0%	8,586ms	7.5
Outline + Tools (No Context Upfront)					
Claude Opus 4	outline_no_ctx	42.0%	36.0%	51,203ms	11.3
Claude Sonnet 4.5	outline_no_ctx	40.0%	34.0%	54,868ms	12.3
Claude Haiku 4.5	outline_no_ctx	32.0%	28.0%	28,631ms	13.1
GPT-4.1	outline_no_ctx	26.0%	10.0%	17,282ms	12.3
GPT-5	outline_no_ctx	26.0%	8.0%	12,517ms	10.3
GPT-4o	outline_no_ctx	22.0%	2.0%	14,639ms	12.5

Table 8: Complete DocFinQA results for all 36 model $\times$ strategy configurations on 50 test examples. Metrics: Accuracy (LLM judge), Numeric Match (exact within 1%), Latency (end-to-end ms), and average tool calls per question. Claude Sonnet 4.5 with RAG+Tools achieves the best accuracy (46%) while maintaining moderate tool usage (12.6 calls).

Parameter	Value	RAG	+Tools	Δ
Chunk Size				
	500 tok	4%	42%	+38
	1,000 tok	6%	46%	+40
	2,000 tok	8%	44%	+36
	4,000 tok	10%	40%	+30
Retrieved Chunks (k)				
	3	2%	44%	+42
	5	4%	46%	+42
	10	6%	46%	+40
	20	8%	44%	+36
Max Tool Calls				
	3	-	28%	-
	5	-	36%	-
	10	-	42%	-
	15	-	46%	-
Embedding Model				
	TF-IDF	6%	46%	+40
	emb-3-small	8%	48%	+40
	emb-3-large	10%	48%	+38

Table 9: Hyperparameter sensitivity analysis on DocFinQA using Claude Sonnet 4.5. Each row shows RAG-only accuracy, RAG+Tools accuracy, and the improvement (Δ). Optimal settings: 1,000-token chunks, k=10 retrieved, 15 max tool calls. Key insight: tools compensate for suboptimal chunking/retrieval—even k=3 with tools (+42) outperforms k=20 without tools (8%).

Example Errors. *Missing context (RAG-only):* "What was YoY change in OpEx?"-retrieved 2019 but not 2018 data, model abstained. *Calculation (RAG+Tools):* Model retrieved correct values ($4.2B/$12.8B) but computed 32.8% instead of 32.81%. *Wrong table:* The model read "Quarterly Results" instead of "Annual Summary," returning Q4 instead of full-year net income.

E Qualitative Examples

Successful Tool Usage. Q: "What was the percentage increase in R&D spending from 2018 to 2019?" Tool trajectory: (1) `search("R&D 2018 2019")` → section 7.2, (2) `read ("7.2")` → expense table, (3) `search ("research development")` → confirms terminology, (4) Answer: "R&D increased from $2.1B to $2.8B, a 33.3% increase." The model uses search to locate, read to retrieve, and verify with additional search.

Failure Case: Over-Exploration. Q: "How many employees does the company have?" Tool trajectory: `search("employees")` → 8 results, then reads sections 1.1, 5.3, 8.1, and 6 more before hitting tool limit without synthesizing an answer. The simple factoid could have been answered from the first search result.

Error Type	RAG Only	RAG+Tools
Missing context	62%	18%
Calculation error	12%	34%
Wrong table/row	14%	22%
Format mismatch	8%	16%
Hallucination	4%	10%

Table 10: Error distribution analysis (n=100 errors examined). With tools, "missing context" errors drop dramatically (62%→18%) as models can request additional information, but calculation errors increase (12%→34%) as models attempt more complex multi-step reasoning.

F Prompt Templates

Base System Prompt. "You are a financial document analyst. Answer questions accurately based on document content. Extract information carefully, perform calculations step-by-step, and state final answers clearly. Format: percentages as X.X%, currency as $X.XM/B, ratios as X.XX."

Tool-Augmented Addition. "You have document navigation tools: search(query), read(id), preview(id), list(). Plan exploration: start with search/list, read promising sections, verify, and synthesize."

No-Context-Upfront. "Document content is NOT provided upfront. You MUST use tools: start with get_outline()/list(), search keywords, read sections, and build understanding iteratively. Do not guess."