

OfficeQA Pro: An Enterprise Benchmark for End-to-End Grounded Reasoning

Databricks AI Research^{1,*}

¹Databricks

*A complete list of contributors is given in the appendix.

We introduce OfficeQA Pro, a benchmark for evaluating AI agents on grounded, multi-document reasoning over a large and heterogeneous document corpus. The corpus consists of U.S. Treasury Bulletins spanning nearly 100 years, comprising 89,000 pages and over 26 million numerical values. OfficeQA Pro consists of 133 questions that require precise document parsing, retrieval, and analytical reasoning across both unstructured text and tabular data. Frontier LLMs including Claude Opus 4.6, GPT-5.4, and Gemini 3.1 Pro Preview achieve less than 5% accuracy on OfficeQA Pro when relying on parametric knowledge, and less than 12% with additional access to the web. When provided directly with the document corpus, frontier agents still struggle on over half of questions, scoring 34.1% on average. We find that providing agents with a structured document representation produced by Databricks' `ai_parse_document` yields a 16.1% average relative performance gain across agents. We conduct additional ablations to study the effects of model selection, table representation, retrieval strategy, and test-time scaling on performance. Despite these improvements, significant headroom remains before agents can be considered reliable at enterprise-grade grounded reasoning.

Date: March 2026

Code: <https://github.com/databricks/officeqa>



1 Introduction

Agent Performance on OfficeQA

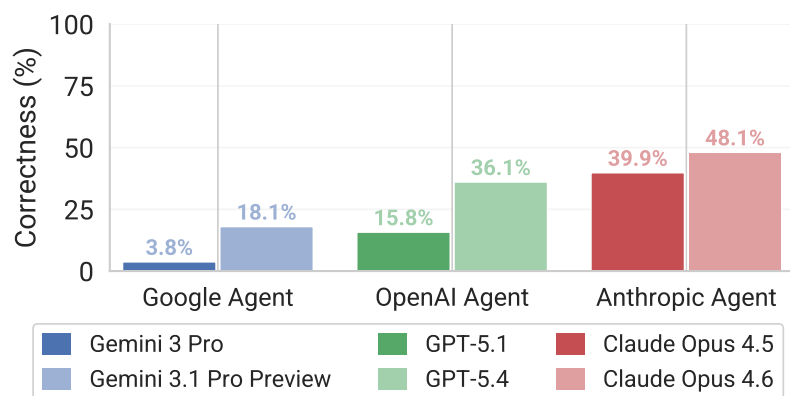


Figure 1 Agent performance on OfficeQA Pro. Each agent framework—Google Agent (Gemini CLI), OpenAI Agent (Codex CLI), and Anthropic Agent (Claude Agent SDK)—is evaluated with the provider's latest flagship model, along with the flagship model available at the time OfficeQA was created. Agent performance with latest frontier models is 34.1% on average and none of the agents surpass 50% accuracy.

Recent benchmarks such as Humanity's Last Exam (HLE) (Phan et al., 2025) and ARC-AGI-2 (Chollet et al., 2026) probe frontier reasoning ability, but often remain decoupled from practical tasks encountered in enterprise environments. GDPval (Patwardhan et al., 2025) represents an important step towards evaluating economically valuable tasks, but does not fully capture the demands of real-world enterprise workflows. Tasks are designed for a closed-world setting where the full context (fewer than two artifacts on average) is provided

directly in the prompt, which neglects the challenge of faithful retrieval across large enterprise collections. Real-world tasks require navigating large, heterogeneous document corpora, identifying and retrieving relevant materials and then performing grounded analysis – a task we refer to as *Grounded Reasoning*. Finally, the benchmark relies on human experts for grading tasks, which makes evaluation onerous.

To bridge these gaps, we introduce **OfficeQA Pro** — a verifiable benchmark specifically designed to proxy economically valuable tasks centered on Grounded Reasoning. Unlike existing benchmarks that focus on niche academic knowledge or closed-world tasks, OfficeQA Pro evaluates end-to-end real-world enterprise capabilities of agents across document-intensive workflows, built on an archive spanning a century of U.S. Treasury Bulletins. Answers are also designed to be easily verifiable, and do not require human-expert grading.

While frontier models already excel at specialized Olympiad-style reasoning tasks (Hubert et al., 2025), this proficiency does not translate across economically valuable tasks, such as those proxied by OfficeQA Pro. Frontier models answer OfficeQA Pro questions correctly less than 5% of the time when relying on parametric knowledge. Even with access to the full corpus, state-of-the-art agents with the latest frontier models (Claude Opus 4.6, GPT 5.4, Gemini 3.1 Pro Preview) answer <50% of questions correctly, as shown in Figure 1¹.

In this paper, we introduce OfficeQA Pro and describe the design principles and methodology used to construct it. We evaluate a range of frontier models and agent architectures, including Claude Agent SDK, OpenAI Codex SDK, and Gemini CLI, and compare their performance with human annotators. We further study the impact of document parsing quality, showing that specialized parsing via Databricks’ `ai_parse_document` yields a 16.1% average relative performance gain across agents. Finally, we analyze how agent design choices such as table representation, search strategies, and test-time scaling affect performance.

2 The OfficeQA Pro Benchmark

In this section, we describe the design principles behind OfficeQA Pro, the process used to create the benchmark, its final composition, and the evaluation methodology used to measure performance.

2.1 Dataset Desiderata

1. **Emulating enterprise data complexity:** The dataset must capture the full diversity of real-world physical documents, which involves various data formats such as prose and tabular information, data drift over long periods, and the noise and scale of real-world archives.
2. **Evaluating multi-step reasoning:** The benchmark must test multi-step reasoning that extends beyond parametric knowledge and forces systems to navigate to the correct documents and sections, extract relevant evidence while considering context, and compute values.
3. **Supporting high-precision, verifiable evaluation:** Questions must have a single unambiguous ground truth and enforce strict correctness, mirroring the precision required in real-world financial workflows and enabling straightforward, automated evaluation.

2.2 Benchmark Composition

To achieve these goals, we constructed OfficeQA Pro: a large-scale benchmark designed to mirror the structural complexity of enterprise document workflows, requiring models to locate and reason over information across hundreds of reports spanning multiple decades. The OfficeQA Pro corpus consists of U.S. Treasury Bulletins, which were published monthly from 1939-1982 and quarterly thereafter.

Each bulletin is 100–200 pages long and consists of written analysis, numerical tables, figures, and charts. Tabular data in the bulletins can be highly complex, with multiple levels of nested hierarchy, changing units, and extensive footnotes. Published statistics are often revised in subsequent months and years, requiring longitudinal reconciliation. Additionally, the corpus transitions from scans of physical documents to digital-native PDFs in 1996, introducing variability in character recognition quality and layout consistency. Many PDFs include an embedded text layer, which we remove to ensure fair assessment of document interpretability.

¹Performance of GPT 5.1 and Claude Opus 4.5 Agents differ from what was originally reported in the [OfficeQA blog](#) due to using updated agent architectures, reporting on OfficeQA Pro vs. Full, as well as revisions to a subset of questions.

the new annotator’s answer differed from the original, the sample was sent for a third annotator to review.

After initial verification, we ran two rounds of end-to-end quality assurance. Reviewers inspected questions where AI agents produced conflicting answers and evaluated each discrepancy to understand if (1) the alternative answer came from a failure mode of the agent, (2) the alternative answer was equally correct under the current phrasing of the question, or (3) the alternative answer was actually correct and the ground truth was not. In case (2) the question was revised to remove ambiguity, and in case (3) the ground truth answer was corrected. The full verification flow is shown in Figure 4.

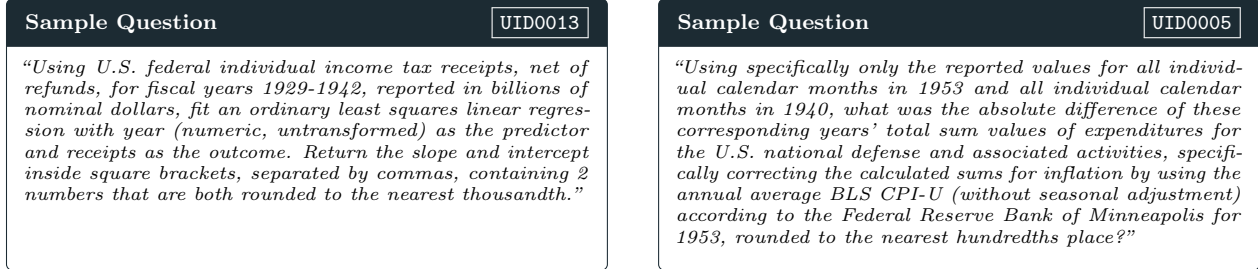


Figure 3 Sample questions from OfficeQA Pro. Question UID0013 (left) requires information from 1 bulletin, and linear regression analysis. Question UID0005 (right) requires information from 2 separate bulletins, multi-step math, and web-search to retrieve the correct CPI value.

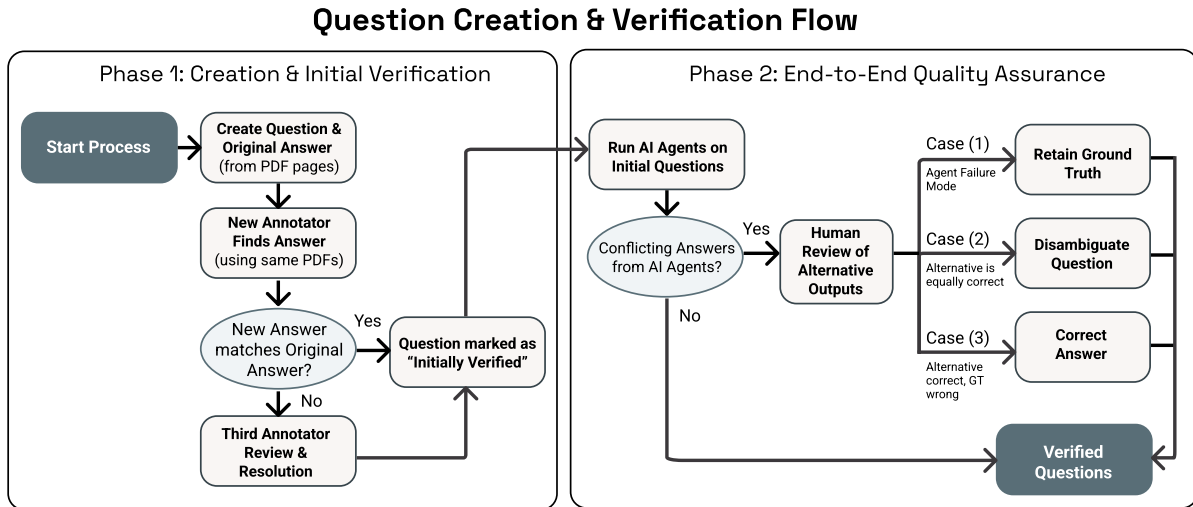


Figure 4 Overview of the benchmark creation process. In Phase 1, the question is created and initially verified by a new annotator. In Phase 2, two passes of end-to-end quality assurance are completed using AI agents to produce alternative answers, which are used to inform human verification of both the question and the ground truth answer.

Filtering To ensure that questions required retrieval from the corpus, we filtered out those solvable using frontier models’ parametric knowledge alone². We found that models had memorized a surprising breadth of historical financial statistics. For example, some models could answer “What were the monthly average forward rates implied by U.S. Treasury market bid yields between the 10-year and 30-year maturities in December 1995? Report the value rounded to the nearest hundredth.” without consulting external sources. After this filter, 246 questions remained. We then evaluated them using the strongest agent systems available during benchmark development³. Questions both agents answered correctly were labeled *Easy* and retained only in the OfficeQA Full split, leaving 133 questions in OfficeQA Pro.

²This filtering step used the frontier models available at the time of dataset construction, specifically Claude Opus 4.5 and GPT-5.1. More recent frontier models available at the time of writing, such as Claude Opus 4.6 and GPT-5.4, can answer some of these questions via parametric knowledge that earlier models could not.

³The frontier agents used during benchmark construction were powered by Claude Opus 4.5 and GPT-5.1 and leveraged Databricks Parsed documents to improve performance.

2.4 Evaluation

Each question in OfficeQA Pro has a single unambiguous answer, enabling deterministic evaluation via exact match. Since 99% of answers in the benchmark are numerical, we additionally report accuracy under varying *allowable absolute relative error* thresholds. Given a ground-truth value y and prediction \hat{y} , the absolute relative error is defined as:

$$\text{Absolute Relative Error (\%)} = \left| \frac{\hat{y} - y}{y} \right| \times 100 \quad (1)$$

A prediction \hat{y} is considered correct if its absolute relative error falls within the specified threshold; otherwise it is marked incorrect. For example, if the ground truth is 543 million, a prediction of 543.1 million would be incorrect at a 0.0% allowable error threshold but correct at 0.1%. Unless otherwise stated, we report OfficeQA Pro performance using a 0.0% allowable absolute relative error threshold.

To ensure fairness across diverse answer formats, the evaluation metric normalizes variations in punctuation, mathematical symbols, and common abbreviations. For complex responses such as dates or multi-part answers, correctness requires overlap in both the primary text and numerical components to prevent false matches. Non-numerical responses are evaluated using fuzzy text matching that tolerates minor formatting differences while preserving semantic equivalence. The full reward implementation is available [here](#).

3 Frontier AI Performance on OfficeQA Pro

In this section, we evaluate how frontier AI systems, including both LLMs and agent-based approaches, perform on OfficeQA Pro. We define *LLM baselines* as systems consisting of a single call to a frontier model, optionally augmented with additional context or web search capabilities. *Agent baselines* refer to orchestration frameworks that enable autonomous tool use and multi-step reasoning by the underlying LLM. In addition to the AI baselines described in this section, human performance baselines are also presented in Appendix C.

3.1 LLM Baselines

To understand how frontier LLMs perform on OfficeQA Pro, we evaluate models from OpenAI (GPT 5.4 - high reasoning⁴), Anthropic (Claude Opus 4.6) and Google (Gemini 3.1 Pro Preview) using the configurations⁵:

- **Prompt Only:** LLMs answer with the question alone, and no documents, tools, or external context. This isolates the models’ parametric knowledge and provides a lower bound for performance.
- **Web Search Enabled:** LLMs can access the internet through their native web search tool capabilities. This tests how well models locate the Treasury Bulletin information available online.
- **Oracle PDF Page(s) + Web Search:** LLMs are provided directly with the oracle PDF page(s) required for answering the question. When a question spans multiple PDFs, the relevant pages containing the referenced information are concatenated as the composite oracle input. This allows us to evaluate model performance under oracle retrieval settings while additionally enabling the web search tool for LLMs, allowing models to retrieve any external information needed for some questions.
- **Oracle Parsed Page(s) + Web Search:** To isolate the effect of document parsing of PDFs on the downstream accuracy, LLMs are presented with the oracle page(s) after pre-parsing the PDFs using Databricks’ `ai_parse_document`, which extracts document content into a structured representation that encodes text, tables and other relevant information.⁶

System prompts for each configuration are provided in E.4. We plot accuracy for all configurations in Figure 5 with full results including average latency in Appendix E.1. In the prompt-only setting, every model performs poorly at under 3% accuracy at 0.0% allowable absolute relative error. Interestingly, models demonstrate an ability to make reasonable Fermi estimates, as performance increases to 17-24% at 5% allowable absolute relative error. Traces suggest that models often use data from internalized government sources but frequently

⁴GPT 5.4 was configured to run with high reasoning because Claude Opus 4.6 uses high reasoning by default and Gemini 3.1 Pro Preview uses high thinking level by default.

⁵For each setting, we configure the models to have 50k maximum output tokens.

⁶Because parsed outputs include additional layout metadata such as bounding boxes, we extract the text content from each page element and concatenate it in reading order to produce a simplified .txt representation of each PDF. This transformation is applied to the parsed-document corpus in all experiments throughout the paper.

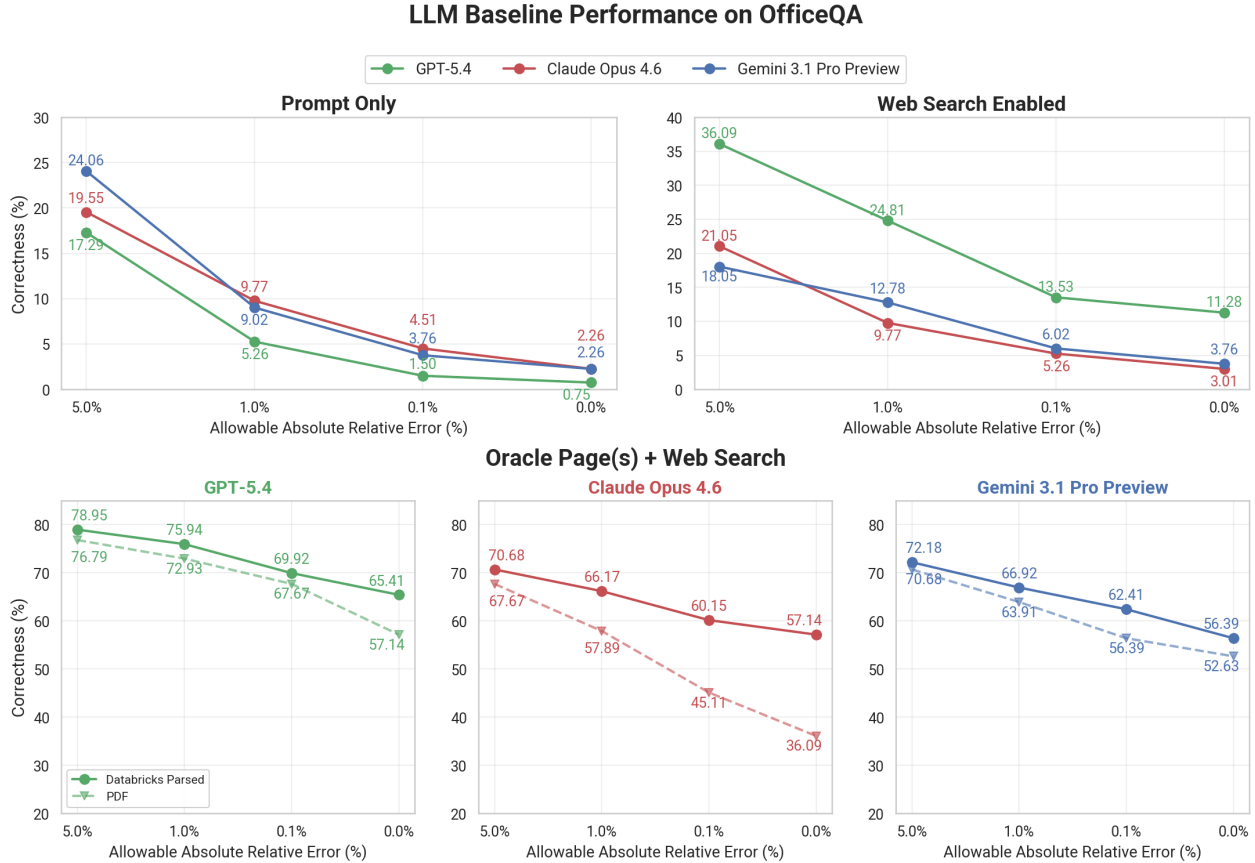


Figure 5 Performance of frontier LLMs on OfficeQA Pro across three configurations: Prompt Only, Web Search Enabled, and Oracle Page(s) Provided + Web Search. Results are reported as correctness (%) across allowable absolute relative error thresholds of 5.0%, 1.0%, 0.1%, and 0.0% for the three models. The Oracle configuration denotes the original documents as “PDF” while “Databricks Parsed” refers to parsing via Databricks’ `ai_parse_document`. Leveraging the Databricks’ parsed pages provides a relative gain of +50.2% on average at 0.0% allowable absolute relative error.

cite values in different contexts or misremember latest data. For instance, on a federal debt ratio question, models referenced Office of Management and Budget figures instead of the Treasury Bulletin and applied the wrong debt definition, yielding approximately 0.5% deviation, equating to a difference of tens of millions of dollars. Additionally, models recognize their knowledge limits, refusing to answer 10-15% of questions.

Enabling web search substantially improves performance, with GPT-5.4 reaching 11.3% accuracy compared to less than 1% in the prompt-only setting. However, many incorrect responses arise from limitations in completing multi-step reasoning within a single generation. Concretely, Opus 4.6 fails to produce a final answer in ~80% of cases, exhausting generation tokens before completing search and reasoning, as does Gemini 3.1 Pro Preview in 15% and GPT-5.4 in 2%. Additionally, the models exhibit similar behavior as the prompt-only baseline in misapplying adjacent terminology or extracting plausible yet incorrect values from different sources. Despite referencing sources that mention U.S. Treasury Bulletin data such as the FRASER Data API (Federal Reserve Bank of St. Louis, 2026) in ~70% of reasoning traces for Opus 4.6, 50% for GPT-5.4 and 5% for Gemini 3.1 Pro Preview, models still extract and apply data incorrectly.⁷

These findings suggest that single-turn LLM interactions are poorly suited to the search and reasoning capabilities OfficeQA Pro requires. We next evaluate an Oracle Pages plus Web Search setting, where models receive the oracle page(s) needed for the question and can use web search for external values. Oracle Pages are provided in two forms: original PDFs and pre-parsed documents via Databricks’ `ai_parse_document`.

With raw PDFs, models achieve 36-57% accuracy at 0.0% allowable absolute relative error. With Databricks parsed documents, we observe gains across models: GPT-5.4 improves from 57.1% to 65.4% (+8.3pp), Claude

⁷Interestingly, none of the models reference the publicly available dataset hosted on GitHub which contains the exact questions, answers and ground truth source documents and links. While an acknowledged risk, this also highlights limitations of the models’ search capabilities.

Opus 4.6 from 36.1% to 57.1% (+21.0pp), while Gemini 3.1 Pro Preview improves from 52.6% to 56.4% (+3.8pp). The raw PDF gaps are driven by extraction errors such as misreading nested multi-table layouts, and incorrectly parsing values from older scanned documents. GPT 5.4 performs $\sim 15\%$ worse on questions involving documents from the 1930s-1970s compared to the more digitally native PDFs from 1980s-2020s, while Opus 4.6 and Gemini 3.1 Pro Preview show a smaller $\sim 5\%$ gap across these periods. Models also exhibit reasoning failures: applying incorrect financial formulas, misaligning units or misinterpreting visual figures. Despite providing oracle pages with web search and high-quality parsing, substantial headroom of $\sim 35\text{-}44\%$ remains under the best configuration across models.

3.2 Agent Baselines

Agent	Corpus	Format	Correctness (%)	Latency (min)	Tool Calls	Cost (\$) ⁸
Claude Opus 4.6	Full	PDF	48.12	31.2	82.4	4.55
		DBX Parsed	54.14	5.3	68.5	5.93
	Oracle	PDF	60.90	7.3	26.5	1.19
		DBX Parsed	66.92	3.5	24.5	2.76
GPT-5.4	Full	PDF	36.09	13.1	57.0	1.79
		DBX Parsed	56.39	3.6	34.5	1.26
	Oracle	PDF	54.89	4.4	31.6	0.44
		DBX Parsed	65.41	2.2	20.7	0.33
Gemini 3.1 Pro Preview	Full	PDF	18.05	26.4	75.2	6.21
		DBX Parsed	29.32	2.9	12.8	1.61
	Oracle	PDF	39.10	4.2	28.4	0.76
		DBX Parsed	46.62	2.6	11.4	0.23

Table 1 Agent performance across evaluated configurations including the Full Corpus vs. Oracle Pages (Full vs. Oracle); and PDFs vs. documents parsed with Databricks’ `ai_parse_document` (PDF vs. DBX Parsed). We report correctness (%) at 0.0% allowable absolute relative error, average latency, average tool calls, and cost per sample.

Next, we evaluate agent baselines implemented using publicly-available agent frameworks from the same set of model providers: Codex CLI (OpenAI), Claude Agent SDK (Anthropic), and Gemini CLI (Google). All agents are run with their respective frontier models and are provided with full access to their native tool ecosystems run in non-interactive modes. These tools include and are not limited to file search, shell command execution, code interpreters, and web search. Questions are processed independently within an isolated virtual environment so that an agent importing a new package does not impact performance on other questions. All agents also share a common system prompt (Appendix E.3).

We test out the following configurations when evaluating our agents:

1. **Oracle Page(s) vs. Full Corpus.** To measure the retrieval bottleneck, we compare performance when agents are given the full corpus versus the oracle page(s) needed to answer the question.
2. **PDFs vs. Parsed Documents.** To assess the impact of document parsing, agents are evaluated on the original PDF documents (which must be parsed internally by the agent) and on pre-parsed documents generated using Databricks’ `ai_parse_document`.

In pilot runs, agents spent most latency and tool calls installing PDF parsing libraries, often taking hours to answer questions. To avoid this overhead, we preinstall a suite of PDF parsing and OCR dependencies gathered from earlier runs (full list in Appendix E.3). The parsed text setting does not require these packages, relying on native Unix command line tools, with any additional installs left to the agent. None of the agents attempt to parse PDFs using their own model capabilities, relying only on tools or external libraries.

Results are shown in Table 1 and Figure 6. Among full-corpus PDF configurations, Claude Opus 4.6 achieves the highest accuracy at 48.1%, followed by GPT-5.4 High (36.1%) and Gemini 3.1 Pro Preview (18.1%). The Claude Opus 4.6 agent makes $\sim 11\text{k}$ tool calls, with 65% using Claude Code’s native Read tool to extract values across pages. In contrast, Codex with GPT-5.4 operates primarily through Bash, with $\sim 90\%$ of tool calls leveraging OCR and PDF CLI tools. The GPT-5.4 agent relies on a `tesseract + pdftoppm + Pillow`

⁸Agents using PDFs would incur significantly higher costs and latencies without the preinstalled parsing dependencies (Appendix E.3); pilot runs without preinstallation resulted in 2–4 hour latencies and 2–3 \times the cost per question.

Agent Baseline Performance on OfficeQA

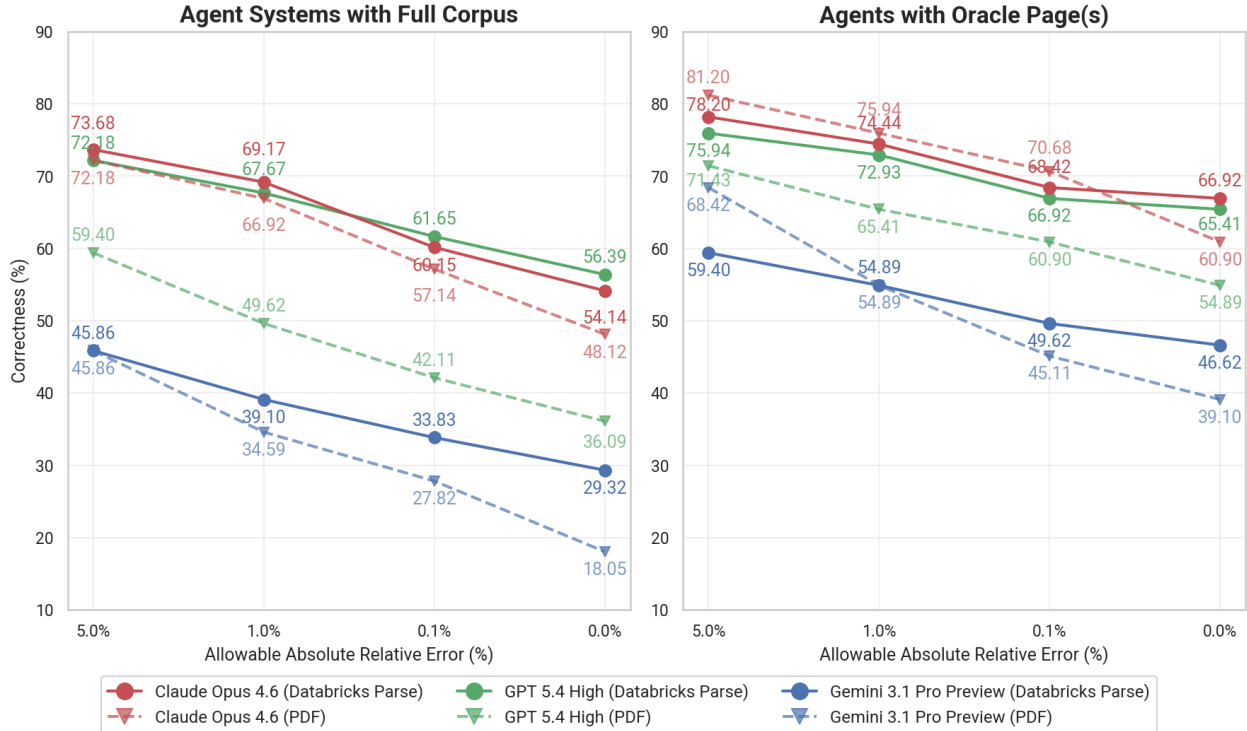


Figure 6 Agent correctness on OfficeQA Pro (left: full corpus; right: oracle pages) across allowable absolute relative error thresholds (5.0%, 1.0%, 0.1%, 0.0%). Solid lines (“Databricks Parse”) use documents parsed via Databricks’ `ai_parse_document`; dashed lines (“PDF”) use the raw PDF corpus.

OCR pipeline, brute-forcing page-by-page extraction rather than text-based PDF parsing. Despite this, the GPT-5.4 agent is still $\sim 2.4\times$ faster than the Claude agent, because it batches OCR extraction into shell one-liners that process many pages locally without API overhead.

When presented with the Databricks’ parsed documents, agent correctness improves by 6.0-20.3 absolute percentage points (12.5–62.4% relative) over PDF corpus baselines. In addition to being more correct, agents with these parsed documents are also 4-9x faster. Similarly, the GPT-5.4 agent’s cost drops by $\sim 30\%$ while the Opus 4.6 agent’s cost remains comparable as it shifts from many extraction calls to fewer but more token-intensive reasoning steps over the pre-parsed text.

We next evaluate agent performance with oracle PDF page(s) provided directly to the agent, isolating the contribution of document retrieval to overall performance. Under this setting, accuracy improves by 13–21 absolute percentage points compared to full-corpus PDF runs. Average latency decreases by approximately 76%, and costs fall by 74–88% across agents. Despite the dramatic reduction in search space, latency improves less than might be expected, as most remaining runtime is still dominated by per-document PDF extraction overhead, including repeated file reads and OCR attempts.

When agents are provided with oracle Databricks’ parsed documents, we observe the best overall performance for all agents. In this setting, correctness improves by 6–11 percentage points for the Opus 4.6 and GPT-5.4 agents relative to oracle PDF inputs. The Claude Opus 4.6 agent achieves the highest overall score, reaching 66.9% accuracy. All agents in this configuration also reflect the lowest latencies and tool calls used, demonstrating the benefit of leveraging parsing capabilities. They are also the most cost-efficient, with the GPT-5.4 agent at \$0.33 per sample, approximately 82% cheaper than the full-corpus PDF configuration.

A majority of remaining errors stem from final answer precision (ex. rounding too early) or applying incorrect formulas for more challenging statistical questions. Other failure modes include retrieving the wrong value from the column-row attribute or an incorrect value from web search for a historical statistic. Finally, figures were omitted from the parsed representations, which expectedly led to agents struggling on such questions involving visual understanding. We leave exploration of encoding strategies for visual figures for future work.

4 Custom Agent Experiments

Next, we evaluate how different components of agent systems impact performance. We conduct these studies using a custom agent, which enables controlled evaluation of individual design decisions. The base agent architecture used in these experiments is described in Section 4.1, and parser selection is examined in Section 4.2. Additional ablations exploring model choice, table representation, retrieval tools, and test-time compute scaling strategies are presented in Appendix D.

4.1 Base Agent Design

The agent is allowed up to 200 steps per question and uses a sliding window that retains the 30 most recent messages, along with a reminder message of remaining steps. The LLM used for the agent backend is responsible for using tools, reasoning about next steps, and producing the final answer.

We provide all agents with the following tools⁹ by default:

1. **Web-Search:** A web search tool using the DuckDuckGo Search API, enabling the agent to query the open internet for supplementary information.
2. **Python Execution:** The Python REPL tool provides the agent with a persistent, stateful Python execution environment for performing numerical computations, data manipulation, and multi-step reasoning. It is sandboxed to prevent bulk file scanning (e.g., `glob`, `os.listdir`, `os.walk`) over the corpus.
3. **File Search:** Custom `fs_search` and `fs_read` tools, which allow the agent to perform basic file search actions similar to `grep`, `sed`, `cat`, and `ls` (mappings highlighted in Table 3). These custom classes include functionality to ensure that the agent cannot read outside of its current working directory, and doesn't read large outputs (e.g., large files potentially returned by `fs_read`).

4.2 PDF Parser Selection

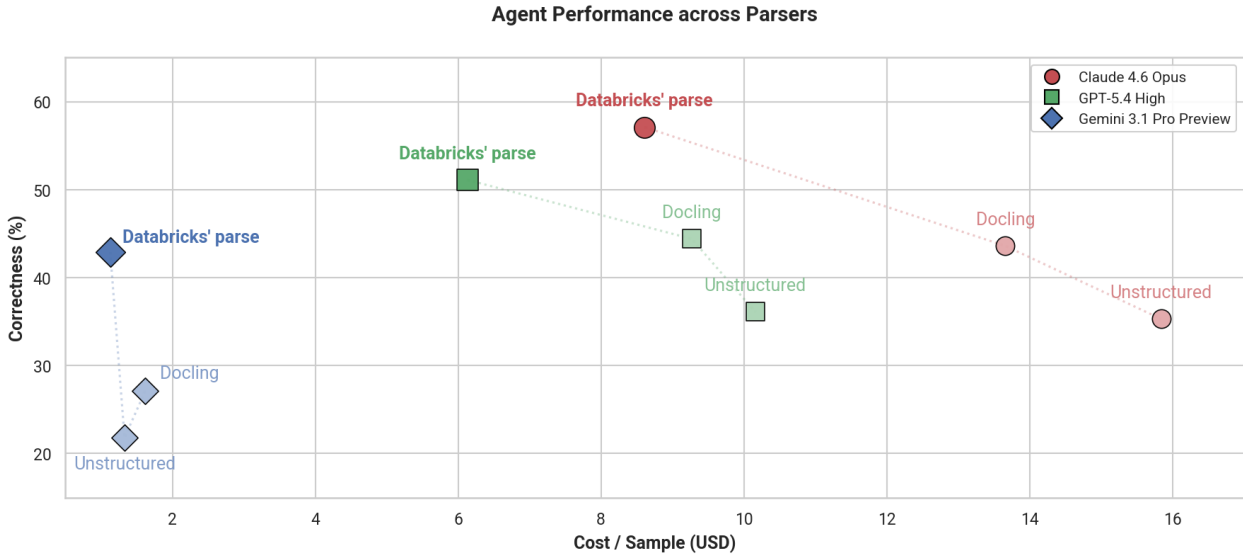


Figure 7 Custom agent correctness (%) at 0.0% error threshold vs. average agent cost per sample on the full OfficeQA Pro corpus using file search, across 3 models and 3 parsers. Databricks' `ai_parse_document` is highlighted in bold.

As shown in the baseline experiments, parsing quality can significantly affect agent performance. We therefore compare three parsing setups to determine which performs best:

1. **Docling:** Open-source PDF document conversion library, using RapidOCR and TableFormer settings.
2. **unstructured.io:** Proprietary high-resolution document parser, costing \$2,670 for full corpus parsing.
3. **Databricks' `ai_parse_document`:** Databricks AI function, costing \$178 for full corpus parsing.

⁹To prevent out of context errors, tool outputs are truncated at 25k characters. All agents are run with a script that allows up to 30 retries to restart the agent from its last attempted sample if the agent crashes or times out during the run.

The parsers’ outputs contain potentially extraneous information, such as bounding boxes that encode the texts’ location on pages. To simplify these inputs, we extract text from each page element and concatenate this together, producing a .txt file for each PDF containing concatenated text appearing in reading order.

To understand which parser achieves the best performance on OfficeQA Pro, we run the custom agent (using 3 different frontier models: GPT 5.4, Claude Opus 4.6, and Gemini 3.1 Pro Preview) using the output corpus from each parser. The resulting accuracies and costs across different parsers are shown in Figure 7. `ai_parse_document` consistently achieves the highest correctness, yielding an average accuracy of 50.4% compared to Docling (38.4%) and unstructured.io (31.1%) across the agent configurations. Additionally, it is most cost-effective with an average cost of \$5.29 per sample, a 35% reduction compared to the next-cheapest option (Docling at \$8.18). Since we find that `ai_parse_document` delivers the best overall agent performance and cost, we proceed with its parsed corpus for the remainder of custom agent experiments in Appendix D.

In these additional experiments, we conduct extensive ablation studies on agent design decisions. Across 10 frontier LLMs (Appendix D.2), Claude Opus 4.6 achieves the highest accuracy (57.1%). While Anthropic models dominate the latency pareto frontier; OpenAI models offer competitive cost efficiency due to lower per-token pricing. Comparing HTML vs. hierarchical Markdown table representations (Appendix D.3), HTML generally improves correctness by a slight margin. Evaluating search tool configurations (Appendix D.4), we find that combining file search with contextual-embedding vector search provides the best quality–cost tradeoff. Finally, test-time scaling via plurality voting (Appendix D.5) yields modest but consistent gains.

5 Remaining Failure Modes

Considerable headroom remains for OfficeQA Pro. In this section, we analyze common failure modes.

5.1 Temporal Revision Verification

Much like enterprise document collections, the U.S. Treasury Bulletins form a continuously updated archive. Reported values may first appear as preliminary estimates and later be revised as additional accounting information becomes available or adjustments are applied. The same fiscal year statistic can recur across multiple documents with differing values due to successive revisions (e.g. Figure 8). Reporting conventions can also drift over time, shifting where values appear and how they are aggregated. Agents frequently fail to retrieve revised values, often prematurely converging on the first numerically plausible instance retrieved, even when explicitly instructed to identify the most recently published figures. Attempts to enforce revision checking often trigger repeated search iterations, which expand context usage and lead to recursive retrieval loops. As the context window saturates, agents lose track of previously validated intermediate results, reverting to approximations when step limits are reached, resulting in cascading errors.

Type of Liability	Calendar Year	2012			2013			
	2011	Dec. r	Jan. r	Feb. r	Mar. r	Apr.	May	June p
Total liabilities to all foreigners.....	4,914,933	4,521,289	4,592,556	4,652,189	4,685,608	4,750,089	4,758,916	4,764,858

Type of Liability	Calendar Year	2013						
	2012 r	Mar. r	Apr. r	May r	June	July	Aug.	Sept. p
Total liabilities to all foreigners.....	4,526,138	4,691,810	4,759,089	4,769,878	4,790,425	4,789,846	4,770,834	4,687,295

Figure 8 Top: "Liabilities to Foreigners" values from the September 2013 Treasury Bulletin (March, May, June 2013). Bottom: revised values for the three months in the December 2013 issue.

5.2 Parsing Faithfulness

Agent performance is highly sensitive to parsing quality. Baseline agents using the original PDFs exhibit a 40–50% failure rate attributable to parsing errors such as misread numbers (see Figure 9), corrupted text, and misaligned tables. Even best-performing agents leveraging state-of-the-art Databricks’ `ai_parse_document` function are impacted by parsing fidelity errors, causing degradation in retrieval and subsequent calculation. Table topology failures also create edge cases like shifted or missing rows and columns, and lost metadata in subheaders, descriptions, and footnotes.

Budget expenditures 2/													
1940.....	677	624	798	710	605	884	843	801	738	891	888	1,187	9,645
1941.....	1,133	1,076	1,482	1,404	1,352	1,591	1,631	1,668	1,989	2,328	1,890	2,684	20,229
1942.....	2,680	2,652	3,515	3,939	4,400	4,810	5,257	5,456	5,921	6,184	6,012	6,926	57,751
1943.....	6,728	6,358	7,746	7,300	7,475	8,260	7,499	7,831	7,667	7,619	7,947	7,744	90,174

Budget expenditures 2/													
1940	677	624	798	710	605	884	843	801	738	891	888	1,187	9,645
1941	1,133	1,076	1,482	1,404	1,352	1,591	1,531	1,668	1,989	2,328	1,890	2,684	20,229
1942	2,680	2,652	3,515	3,939	4,100	4,810	5,257	5,456	5,921	6,184	6,012	6,926	57,751
1943	6,728	6,358	7,746	7,300	7,475	8,260	7,499	7,831	7,667	7,619	7,947	7,744	90,174

Figure 9 Top: original February 1950 Treasury Bulletin table on budget expenditures from CY January 1940 - December 1943. Bottom: parsed output with two errors highlighted (May 1942, July 1941).

5.3 Visual Understanding

Agents frequently fail to correctly utilize visual figures in their analysis. When provided directly with the image (like the one shown in Figure 10), agent systems lack the granular resolution to correctly interpret dense financial charts and struggle to map visual trend lines with high precision. When relying on pre-parsed documents, the retrieval pipeline also faces the bottleneck of a lossy embedding conversion that often dissociates the visual data from its semantic context. This results in agents being unable to effectively retrieve or reason over these modalities, generally leading to ungrounded or inaccurate answers.

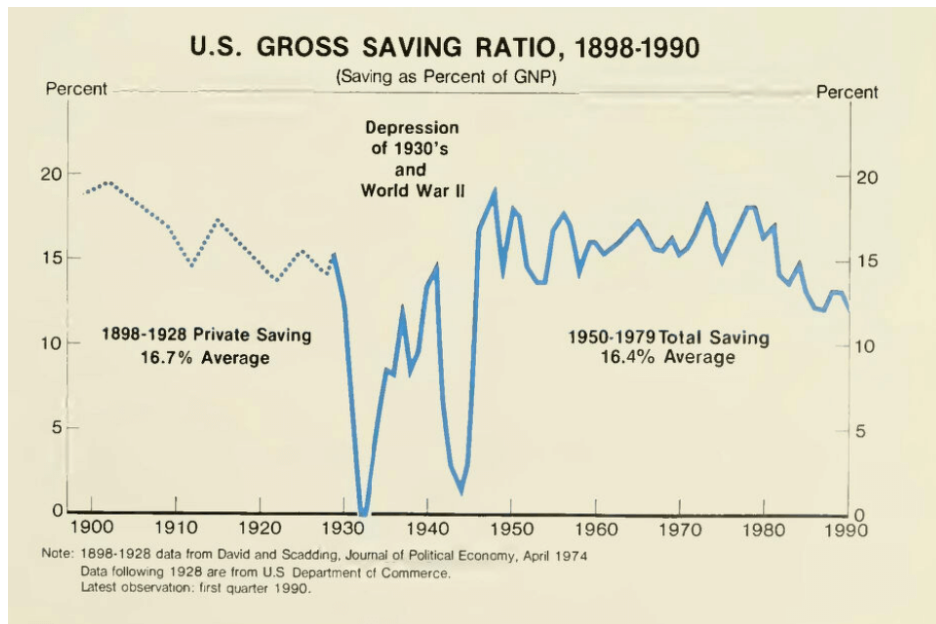


Figure 10 Models struggle with visual reasoning tasks such as determining the total local maxima from this line plot.

5.4 Analytical Reasoning

Analytical reasoning errors occur in multiple forms. First, LLMs and Agent systems are both prone to retrieving values that appear relevant but differ subtly from the quantities required to answer the question. For example, they may use alternative definitions for statistical values or select plausible but imprecise figures from external sources such as CPI values required for inflation adjustments. Mathematical errors also arise as agents apply mathematically valid yet contextually misaligned formulas such as using sample variance instead of population variance. Agents occasionally ignore explicit calculation requirements specified in some questions, such as performing intermediate rounding that results in cascading imprecision. Although less frequent, agents sometimes perform calculations internally instead of delegating them to computational scripts, resulting in avoidable numerical inaccuracies.

6 Related Work

Recent benchmarks have significantly advanced the evaluation of frontier AI capabilities, but largely remain decoupled from enterprise workflows. Humanity’s Last Exam (HLE) (Phan et al., 2025) and ARC-AGI-2 (Chollet et al., 2026) test expert-level academics and abstract intelligence, but offer limited signal on whether such value translates in real-world settings. To bridge abstract reasoning and real-world utility, recent work has explored simulating professional environments for LLMs and agents deployed on domain-specific workflows. GDPVal (Patwardhan et al., 2025) builds tasks from key economic sectors that require agents to produce deliverables such as reports and spreadsheets. APEX-Agents (Vidgen et al., 2026) requires agents to navigate simulated workspace environments for long-horizon tasks like updating financial models or reviewing legal documents. OfficeQA Pro advances this effort by testing agent capabilities over a large heterogeneous document corpus and evaluating rigorous, professionally relevant tasks. Grounded in publicly-available historical U.S. Treasury Bulletins, OfficeQA Pro emulates complex enterprise data and evaluates how AI systems perform on economically valuable real-world workflows.

Grounded reasoning comprises three phases: document parsing, information retrieval and quantitative analysis. Parsing benchmarks such as OmniDocBench (Ouyang et al., 2025) demonstrate how extraction quality degrades on complex PDF layouts across tables, reading order, and dense formatting, which is especially relevant for OfficeQA Pro’s corpus layout variation from pre-1996 scanned documents to modern digital PDFs. Reasoning-intensive retrieval benchmarks such as BRIGHT (Su et al., 2025) or BrowseComp-Plus (Chen et al., 2025) illustrate how standard semantic or keyword matching fails when relevance requires intent reasoning, mirroring OfficeQA Pro where agents must actively identify relevant documents and enclosed sections. Lastly, even with the correct evidence, systems must perform reliable quantitative analysis to produce verifiable numerical answers. FinanceBench (Islam et al., 2023) evaluates open-book financial QA over filings requiring faithful extraction and computation, while long-context benchmarks such as LongBenchv2 (Bai et al., 2025) demonstrate how sustained long-document reasoning remains difficult. OfficeQA Pro unifies these stages, evaluating end-to-end parsing, retrieval and computation within a final verifiable answer.

7 Discussion & Future Work

We present OfficeQA Pro, a benchmark for evaluating AI systems on enterprise grounded reasoning tasks. In our experiments, the strongest frontier agent achieves only 48.1% accuracy. Pre-parsing the corpus with Databricks’ `ai_parse_document` improves performance up to 56.4%, but substantial headroom remains for reliable reasoning over enterprise documents.

Several key findings emerge from our analysis. Document parsing quality stands out as a critical and often underappreciated bottleneck: the choice of parser alone can shift agent accuracy by up to 22 points, and even state-of-the-art parsing errors can propagate through retrieval and downstream computation. In additional experiments in Appendix D, we find that representing tables as HTML over markdown marginally improves correctness, and combining file-based and vector-based retrieval yields a stronger quality–efficiency tradeoff than either method alone. Test-time scaling provides additional gains, though improvements saturate for higher-performing models, suggesting diminishing returns as agent reasoning becomes more consistent. Finally, in Appendix C we find that while AI agents struggle to achieve high performance, they still consistently outperform human annotators in both speed and accuracy.

We observe that substantial headroom remains for both LLM baselines and AI agents. Our failure-mode analysis highlights several directions for future work, including more exhaustive and revision-aware search strategies, stronger visual reasoning over charts and figures, and improved analytical reliability in multi-step quantitative reasoning. Beyond correctness, latency also remains a major and underexplored challenge: across frontier agents, answering a question over the full PDF corpus requires an average of 23.6 minutes, and still 3.9 minutes even when using state-of-the-art document parsing.

We view OfficeQA Pro as an initial step toward a broader class of enterprise evaluations. Future benchmarks should expand across domains, incorporate multiple modalities (e.g., images, databases, and documents), and evaluate tasks that are designed to reflect real-world production workflows across diverse corpora. As an immediate next step, we plan to develop a held-out test set designed to measure agent generalization on grounded reasoning tasks. We hope OfficeQA Pro provides a strong foundation for systematically studying grounded reasoning systems and accelerating progress toward reliable enterprise AI.

References

- Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench v2: Towards deeper understanding and reasoning on realistic long-context multitasks, 2025. <https://arxiv.org/abs/2412.15204>.
- Zijian Chen, Xueguang Ma, Shengyao Zhuang, Ping Nie, Kai Zou, Andrew Liu, Joshua Green, Kshama Patel, Ruoxi Meng, Mingyi Su, Sahel Sharifymoghaddam, Yanxi Li, Haoran Hong, Xinyu Shi, Xuye Liu, Nandan Thakur, Crystina Zhang, Luyu Gao, Wenhu Chen, and Jimmy Lin. Browsecomp-plus: A more fair and transparent evaluation benchmark of deep-research agent. *arXiv preprint arXiv:2508.06600*, 2025.
- Francois Chollet, Mike Knoop, Gregory Kamradt, Bryan Landers, and Henry Pinkard. ARC-AGI-2: A new challenge for frontier ai reasoning systems, 2026. <https://arxiv.org/abs/2505.11831>.
- Federal Reserve Bank of St. Louis. Fraser rest api documentation: /api/author/{authorId}. <https://fraser.stlouisfed.org/api-documentation/rest-api#get-/api/author/-authorId->, 2026. Accessed: 2026-02-24.
- Daniel Ford. Introducing contextual retrieval. <https://www.anthropic.com/engineering/contextual-retrieval>, 2024. Anthropic Engineering Blog.
- Thomas Hubert, Rishi Mehta, Laurent Sartran, Miklós Z. Horváth, Goran Žužić, Eric Wieser, Aja Huang, Julian Schrittwieser, Yannick Schroecker, Hussain Masoom, Ottavia Bertolli, Tom Zahavy, Amol Mandhane, Jessica Yung, Iuliya Beloshapka, Borja Ibarz, Vivek Veeriah, Lei Yu, Oliver Nash, Paul Lezeau, Salvatore Mercuri, Calle Sönne, Bhavik Mehta, Alex Davies, Daniel Zheng, Fabian Pedregosa, Yin Li, Ingrid von Glehn, Mark Rowland, Samuel Albanie, Ameya Velingker, Simon Schmitt, Edward Lockhart, Edward Hughes, Henryk Michalewski, Nicolas Sonnerat, Demis Hassabis, Pushmeet Kohli, and David Silver. Olympiad-level formal mathematical reasoning with reinforcement learning. *Nature*, 2025. doi: 10.1038/s41586-025-09833-y.
- Pranab Islam, Anand Kannappan, Douwe Kiela, Rebecca Qian, Nino Scherrer, and Bertie Vidgen. FinanceBench: A new benchmark for financial question answering, 2023. <https://arxiv.org/abs/2311.11944>.
- Linke Ouyang, Yuan Qu, Hongbin Zhou, Jiawei Zhu, Rui Zhang, Qunshu Lin, Bin Wang, Zhiyuan Zhao, Man Jiang, Xiaomeng Zhao, Jin Shi, Fan Wu, Pei Chu, Minghao Liu, Zhenxiang Li, Chao Xu, Bo Zhang, Botian Shi, Zhongying Tu, and Conghui He. OmniDocBench: Benchmarking diverse pdf document parsing with comprehensive annotations, 2025. <https://arxiv.org/abs/2412.07626>.
- Tejal Patwardhan, Rachel Dias, Elizabeth Proehl, Grace Kim, Michele Wang, Olivia Watkins, Simón Posada Fishman, Marwan Aljubeih, Phoebe Thacker, Laurance Fauconnet, Natalie S. Kim, Patrick Chao, Samuel Miserendino, Gildas Chabot, David Li, Michael Sharman, Alexandra Barr, Amelia Glaese, and Jerry Tworek. Gdpval: Evaluating ai model performance on real-world economically valuable tasks, 2025. <https://arxiv.org/abs/2510.04374>.
- Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Chen Bo Calvin Zhang, Mohamed Shaaban, John Ling, Sean Shi, Michael Choi, Anish Agrawal, Arnav Chopra, Adam Khoja, Ryan Kim, Richard Ren, Jason Hausenloy, Oliver Zhang, Mantas Mazeika, Dmitry Dodonov, Tung Nguyen, Jaeho Lee, Daron Anderson, Mikhail Doroshenko, Alun Cennyth Stokes, Mobeen Mahmood, Oleksandr Pokutnyi, Oleg Iskra, Jessica P. Wang, John-Clark Levin, Mstyslav Kazakov, Fiona Feng, Steven Y. Feng, Haoran Zhao, Michael Yu, Varun Gangal, Chelsea Zou, Zihan Wang, Serguei Popov, Robert Gerbicz, Geoff Galgon, Johannes Schmitt, Will Yeadon, Yongki Lee, Scott Sauers, Alvaro Sanchez, Fabian Giska, Marc Roth, Søren Riis, Saiteja Utpala, Noah Burns, Gashaw M. Goshu, Mohinder Maheshbhai Naiya, Chidozie Agu, Zachary Giboney, Antrell Cheatom, Francesco Fournier-Facio, Sarah-Jane Crowson, Lennart Finke, Zerui Cheng, Jennifer Zampese, Ryan G. Hoerr, Mark Nandor, Hyunwoo Park, Tim Gehrunger, Jiaqi Cai, Ben McCarty, Alexis C Garretson, Edwin Taylor, Damien Sileo, Qiuyu Ren, Usman Qazi, Lianghui Li, Jungbae Nam, John B. Wydallis, Pavel Arkhipov, Jack Wei Lun Shi, Aras Bacho, Chris G. Willcocks, Hangrui Cao, Sumeet Motwani, Emily de Oliveira Santos, Johannes Veith, Edward Vendrow, Doru Cojoc, Kengo Zenitani, Joshua Robinson, Longke Tang, Yuqi Li, Joshua Vendrow, Natanael Wildner Fraga, Vladyslav Kuchkin, Andrey Pupasov Maksimov, Pierre Marion, Denis Efremov, Jayson Lynch, Kaiqu Liang, Aleksandar Mikov, Andrew Gritsevskiy, Julien Guillod, et al. Humanity’s last exam, 2025. <https://arxiv.org/abs/2501.14249>.
- Hongjin Su, Howard Yen, Mengzhou Xia, Weijia Shi, Niklas Muennighoff, Han yu Wang, Haisu Liu, Quan Shi, Zachary S. Siegel, Michael Tang, Ruoxi Sun, Jinsung Yoon, Sercan O. Arik, Danqi Chen, and Tao Yu. BRIGHT: A realistic and challenging benchmark for reasoning-intensive retrieval, 2025. <https://arxiv.org/abs/2407.12883>.
- Yuan Sui, Mengyu Zhou, Mingjie Zhou, Shi Han, and Dongmei Zhang. Table meets llm: Can large language models understand structured table data? a benchmark and empirical study. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining (WSDM)*, 2024.
- Bertram Vidgen, Austin Mann, Abby Fennelly, John Stanly, Lucas Rothman, Marco Burstein, Julien Benček, David Ostrofsky, Anirudh Ravichandran, Debnil Sur, Neel Venugopal, Alannah Hsia, Isaac Robinson, Calix Huang, Olivia Varones, Daniyal Khan, Michael Haines, Zach Richards, Chirag Mahapatra, and Osvald Nitski. APEX-Agents, 2026.

A Contributions

A.1 Authors

Krista Opsahl-Ong[†], Arnav Singhvi[†], Jasmine Collins, Ivan Zhou, Cindy Wang, Ashutosh Baheti, Owen Oertell, Jacob Portes, Sam Havens, Erich Elsen, Michael Bendersky, Matei Zaharia, Xing Chen.

[†]Equal contribution; order is alphabetical.

A.2 Acknowledgements

We thank Dipendra Kumar Misra, Andrew Drozdov, Jonathan Chang, Simon Favreau-Lessard, Erik Lindgren, Pallavi Koppol, and Veronica Lyu for contributions including helpful discussions, seed question development, and efforts to ensure dataset quality. We also thank the github community, including Kimi Kong and Diane Tc for their suggestions for revisions that were incorporated in the dataset.

We are grateful to SuperAnnotate and Turing for their support in creating the OfficeQA questions, and to SuperAnnotate for their assistance in designing and running the human annotation studies.

Finally, we thank USAFacts for guidance in identifying the U.S. Treasury Bulletins and for feedback to ensure the questions were topical and relevant.

Correspondence to krista.opsahl-ong@databricks.com & arnav.singhvi@databricks.com.

B OfficeQA Full

OfficeQA Full is an extended version of OfficeQA Pro that contains an additional set of 113 easier questions to enable hillclimbing and evaluation of less powerful LLM and agent systems. Questions were classified as "Easy" if state-of-the-art frontier agents at the time – which used GPT 5.1 and Claude Opus 4.5 and Databricks parsed documents as input – both answered correctly. Figure 11 shows the breakdown of "Easy" questions, versus the "Pro" difficulty questions found in the core OfficeQA Pro benchmark. The right subplot shows how these question types compare in terms of the distribution of capabilities that they test.

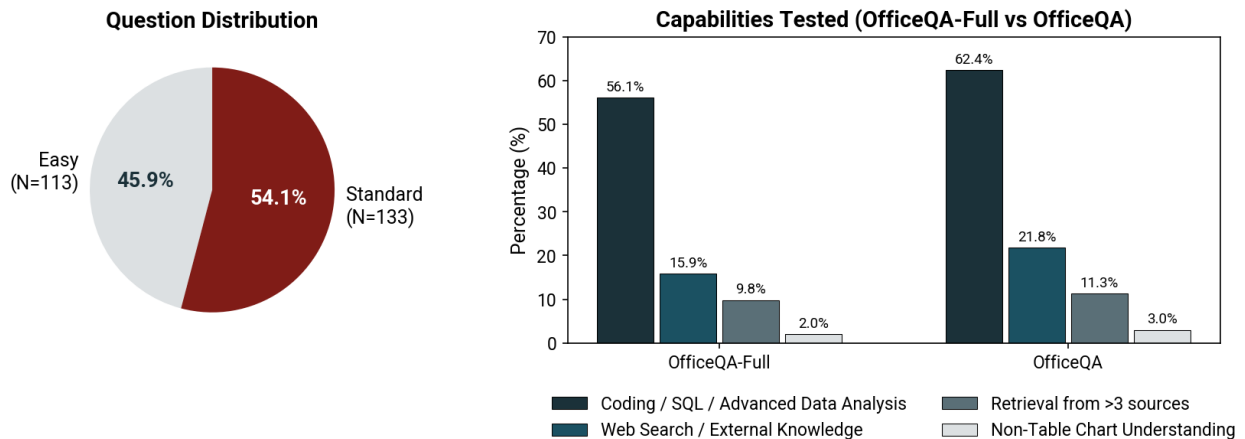


Figure 11 Question difficulty distribution and capability breakdown of OfficeQA-Full and Pro. **Left** shows the difficulty distribution of questions (113 easy / 133 hard). **Right** details sample breakdown by the capabilities required to answer the question, including data analysis, retrieval from >3 distinct pages, external knowledge search, and visual understanding.

In general, the easier questions in OfficeQA-Full tend to require information from less unique pages across the dataset, and are more likely to involve basic arithmetic (ex. basic sums) as opposed to more complex data analytics. Examples of an easy question, compared with a question from the core OfficeQA Pro benchmark can be seen in Figure 12.



Figure 12 Sample questions from the OfficeQA benchmark, including the Full and Pro split. Questions typically require multi-step analytical reasoning such as statistical modeling across multiple documents and datapoints.

C Human Annotator Experiments

We explore how human annotators perform on OfficeQA Full. For these experiments, we employed three annotators who had previously helped create questions and answers for the benchmark and were broadly familiar with the task and the documents¹⁰. We evaluated human performance on a subset of 30 randomly selected questions under two settings:

- Annotators provided with Oracle Page(s).** In these experiments, annotators were provided with the question along with the exact PDF pages required to answer it. They were also given the ability to perform web searches to identify external information as needed.
- Annotators provided with Full Corpus.** Annotators were provided with the question along with a direct link to the full Treasury Bulletin corpus hosted by FRASER. This interface allowed annotators to explore all Treasury Bulletin PDFs, organized by decade. It also supported basic search functionality (e.g., searching for specific keywords in the corpus or identifying the first page for a section such as “Capital Movements” across documents).

We ran the experiment with the full corpus setting first to ensure that annotators had not seen the oracle pages before needing to search for them independently. Annotators were not told the name of the benchmark to prevent them from searching for answers online. In all experiments, annotators were prompted with the same instructions provided to agents, including reminders to use full numerical precision when calculating final answers and to always use the latest revised value from the Treasury Bulletins unless otherwise specified.

We measure human correctness using the same reward function as in our agent experiments. We measure time to answer completion as the period between when an annotator clicks “start” to reveal a new question and when they press “submit” on their final answer. Annotators were informed that this time was being recorded and were instructed to begin a question only when ready to complete it in a single sitting.

Figure 13 shows performance of each annotator with allowable absolute relative error (%) varied from 0 to 5%, as compared with the best agent performance under the same PDF input setting. Table 2 shows how the average human annotator compares with the average agent performance in terms of both accuracy and speed.

Corpus	Config	Format	Correctness (%)	Latency (min)
Oracle Pages	Human (N=3)	PDF	51.1	19.2
	Agent (N=3)	PDF	71.1	5.3
	Agent (N=3)	Parsed	72.2	2.1
Full Corpus	Human (N=3)	PDF	34.6	31.4
	Agent (N=3)	PDF	50.0	15.4
	Agent (N=3)	Parsed	56.7	3.5

Table 2 Human vs. agent performance on a N=30 subset of the OfficeQA-Full benchmark. Human results span 3 annotators; agent results average Claude Opus 4.6, GPT-5.4, and Gemini 3.1 Pro Preview (N=3).

¹⁰They did not answer any questions they created or reviewed.

Human vs. AI Agent Performance on OfficeQA Subset (N=30)

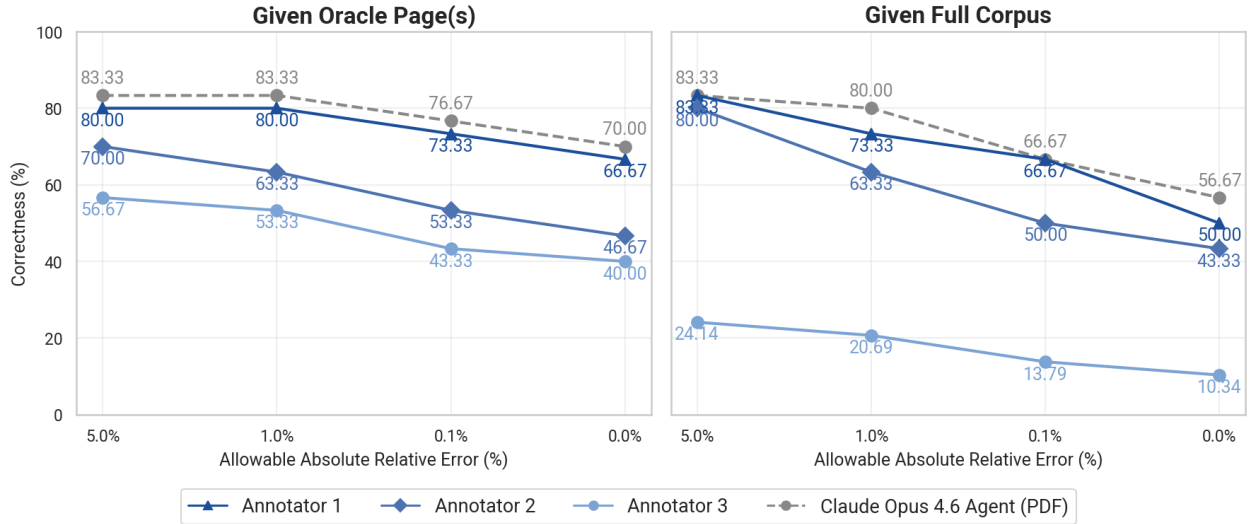


Figure 13 Performance of three human annotators and one AI agent (Claude Opus 4.6, PDF) on a 30-sample subset of OfficeQA-Full across all error thresholds (5.0% to 0.0%). Left: oracle pages provided; right: full corpus.

We find that on average, human annotators generally underperform agent systems in both accuracy and time to completion. When provided directly with oracle pages, humans take $\sim 4x$ longer than the average agent to answer the question, and achieve $\sim 28%$ lower accuracy relative to agent performance at 0.0% allowable absolute relative error. Interestingly, even with access to the oracle PDFs, annotators most often failed by misidentifying the necessary values to perform calculation, usually selecting a figure with subtle differences from what the question was testing (e.g. retrieving a figure from the page on Total “Public Debt Securities” instead of “Total interest-bearing Public Debt”, the value referenced by the question).

Humans also made basic instruction-following mistakes compared to agents, such as answering in incorrect units (e.g., reporting 19.519 trillion instead of 19,519,306 million when the question requests millions), or using an incorrect representation (e.g., a percentage instead of a decimal). While these mistakes may seem trivial, they can shift the final answer by orders of magnitude. A few answers are also incorrect due to transcription typos during intermediate computation or typing in the final response (e.g., answering 0.8525 instead of 0.88525), a failure mode never seen from agents who operate programmatically. On the other hand, when agents operate over the PDFs, they often struggle with parsing the raw document content. Agents can also trigger content policy refusals due to the nature of content provided. Neither of these errors are present from humans who can coherently read the PDFs and always answer the questions. Both failure modes do notably decrease significantly for agents when provided with the parsed documents instead.

When provided with the full corpus of PDFs, average human performance decreases by 16.5 points, which is broadly comparable to the average agent performance that has a 21.1 point decrease with the PDFs corpus and 15.5 point decrease with the parsed documents corpus. In this full corpus setting, agents outperform humans in both accuracy and speed across both document formats. With PDF inputs, agents average $\sim 2x$ faster than humans (15.4 vs. 31.4 minutes) while achieving 44% higher relative accuracy. With parsed documents, agents are $\sim 9x$ faster with 64% higher relative accuracy.

These experiments highlight the differences in the ways grounding differs in humans and agent workflows. For humans, the provided PDFs represent the interface with the ability to ascertain formatting, layout, blurry numbers, etc. directly from the page. Agents instead require the content to be legible for them, relying on a translation layer via parsing, and whatever is lost during that translation is the agent’s reality for the rest of the task. When represented correctly, agent performance is substantially better than humans.

D Custom Agent Ablation Studies

In this section, we perform ablation studies (continued from Section 4) to learn how design decisions such as model selection, table representation, search tools, and test-time scaling techniques impact end to end performance.

D.1 Base Agent Design

For these experiments, we use our own custom agent (initially detailed in Section 4.1) which allows us to ablate variables systematically. By default, unless otherwise specified for the following experiments, this agent is configured tools including websearch for identifying external knowledge, Python REPL for numerical computations, and custom `fs_search` and `fs_read` tools for file search. These file search tools are sandboxed functions with similar capabilities to `grep`, `sed`, `cat`, and `ls`, with specific capabilities detailed further in Table 3. In all of these experiments, the base agent uses Databricks parsed documents as its corpus, since this was the best performing parser evaluated in Section 4.2.

Tool	Action	Example Equivalent Shell Command
fs_search	Directory listing	<code>bash ls</code>
	List files by pattern	<code>bash ls treasury_bulletins/*2013*.txt</code>
	Search directory recursively	<code>bash grep -rn -i -B 2 -A 2 "national saving" treasury_bulletins/</code>
	Search for a pattern 'in specific files	<code>bash grep -rn -i "Individual income taxes" treasury_bulletins/treasury_bulletin_2013*.txt</code>
fs_read	Read first N lines	<code>bash head -n 20 foo.txt</code>
	Read a slice of lines	<code>bash sed -n '101,150p' foo.txt</code>
	Read entire file	<code>bash cat foo.txt</code>

Table 3 Mapping of `fs_search` and `fs_read` tool capabilities to equivalent shell commands.

D.2 Model Performance

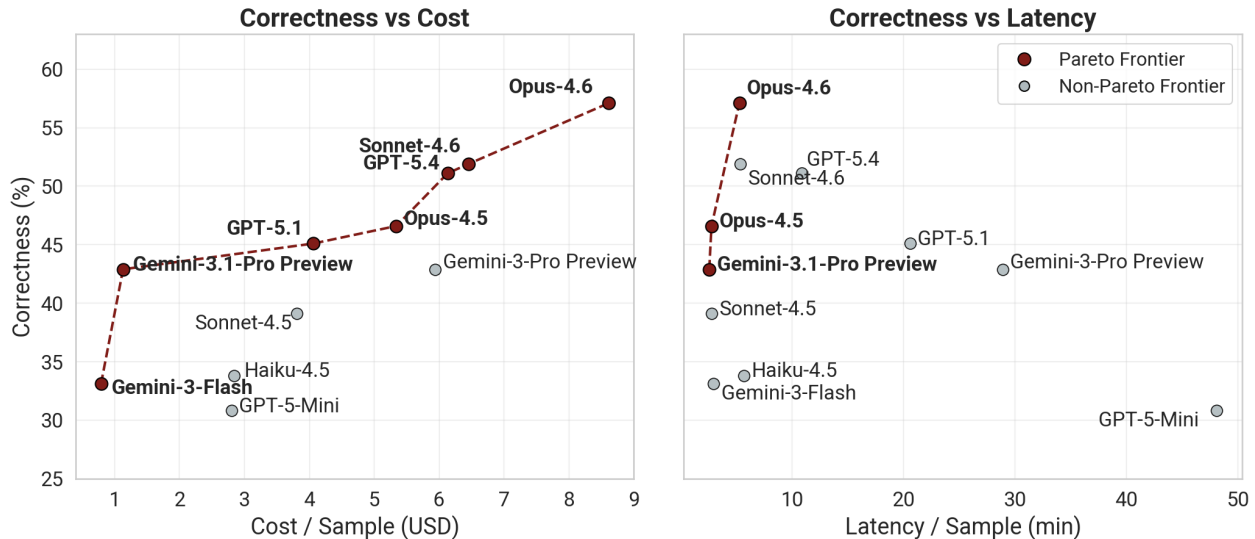


Figure 14 Correctness (%) vs. average cost and latency per sample for the custom agent with Databricks' `ai_parse_document` and file search on the full Treasury Bulletin corpus, evaluated at 0.0% error threshold. Pareto-optimal models are shown in red.

To understand how model selection impacts performance, we evaluate a comprehensive suite of models using the fixed custom agent configuration described above in Section D.1. We evaluate the following models from each provider:

1. **Anthropic:** Claude Opus 4.6, Claude Opus 4.5, Claude Sonnet 4.6, Claude Sonnet 4.5, and Claude Haiku 4.5. We use these models with their out-of-the-box default settings, which uses high reasoning.
2. **OpenAI:** GPT 5.4, GPT 5.1, and GPT 5 Mini. Since the GPT 5 model uses medium reasoning and GPT 5.1 and 5.4 use no reasoning by default, we set them to high reasoning to ensure they’re comparable with Anthropic’s models.
3. **Google:** Gemini 3 Pro Preview, Gemini 3.1 Pro Preview, Gemini 3 Flash. We use these models with their default settings; Google models do not support reasoning settings.

We report agent performance by model in Figure 14 by correctness, cost, and latency profiles. A full table of results including average number of tool calls can also be found in Table 4. We find that using Claude Opus 4.6 yields the best performance, achieving a correctness of 57.1%. While the Claude Opus 4.6 agent also costs the most to run, the average latency required to answer each question was still competitive – lower than about half of the other agents we tested. In general, we find that Anthropic models dominate the pareto-frontier of correctness vs. latency, with the exception of Gemini 3.1 Pro Preview.

This appears to be driven by a few factors. First, Claude Opus and Sonnet models require fewer cumulative tokens to answer questions—for example, Claude Opus 4.6 averages 1.2M total tokens per question compared to 2.3M for GPT 5.4—though this does not hold for Claude Haiku 4.5, which averages 2.2M tokens. Second, Claude models process input tokens $\sim 2.7\times$ faster on average compared with GPT models (5,394 tok/s vs. 1,984 tok/s¹¹).

While Anthropic models dominate in terms of latency, OpenAI models (GPT-5.1-High and GPT-5.4-High) still perform well on the correctness vs. cost pareto frontier. This is largely driven by differences in input token pricing (see Section E.2 for pricing). For example, GPT-5.4’s input cost (\$2.50 / MTok) is approximately 50% cheaper than Claude Opus 4.6 (\$5.00 / MTok) and roughly 17% cheaper than Claude Sonnet 4.5 (\$3.00 / MTok). Because agent workflows in OfficeQA Pro are heavily input-token dominated—due to repeated retrieval, tool usage, and context accumulation—these pricing differences compound substantially at scale. Even when total token usage is higher (e.g., GPT-5.4 averages more turns and total tokens than Claude Opus 4.6), the lower per-token input pricing offsets much of this overhead, resulting in competitive or superior cost-efficiency at similar correctness levels. We note that our pricing estimates do not account for prompt caching discounts, however, we assume our findings would remain directionally similar given that Claude cached input pricing is still $\sim 2x$ higher than GPT models.

Model	Reasoning	Correctness (%)	Latency (min)	Tool Calls	Cost (\$)
GPT-5.4	High	51.10	10.87	104.8	\$6.13
GPT-5.1	High	45.10	20.6	60.1	\$4.06
GPT-5 Mini	High	30.80	48.1	120.4	\$2.80
Claude Opus 4.6	Default	57.10	5.3	53.3	\$8.61
Claude Opus 4.5	Default	46.60	2.8	27.5	\$5.34
Claude Sonnet 4.6	Default	51.90	5.4	69.2	\$6.45
Claude Sonnet 4.5	Default	39.10	2.8	31.2	\$3.81
Claude Haiku 4.5	Default	33.80	5.7	73.9	\$2.84
Gemini 3.1 Pro Preview	Default	42.90	2.6	25.7	\$1.13
Gemini 3 Pro Preview	Default	42.90	28.9	95.1	\$5.94
Gemini 3 Flash	Default	33.10	3.0	40.6	\$0.79

Table 4 Model correctness (%) at 0.0% error threshold, latency (minutes), average tool calls, and cost for the custom agent with Databricks’ `ai_parse_document` and file search on OfficeQA Pro.

¹¹These token/second figures are estimated by dividing the average input tokens per rollout by the average total time the agent takes to answer the question.

D.3 Table Representation

U.S. Currency and Coin Outstanding and in Circulation								
[Source: Financial Management Service]								
AMOUNTS OUTSTANDING AND IN CIRCULATION								
Sept. 30, 1990								
	Total currency and coin	Currency				Coin ²		
		Total	Federal Reserve notes ¹	U.S. notes	Currency no longer issued	Total	Dollars ³	Fractional coin
Amounts outstanding	\$317,134,369,097	\$297,502,055,199	\$296,913,774,209	\$322,539,016	\$265,741,974	\$19,632,313,898	\$2,024,703,898	\$17,607,610,000
Less amounts held by:								
The Treasury	527,367,499	38,918,273	5,803,540	32,906,039	208,694	488,449,226	320,792,133	167,657,093
The Federal Reserve banks . . .	44,709,108,760	44,176,306,805	44,176,287,294	213	19,298	532,801,955	101,462,811	431,339,144
Amounts in circulation	271,897,892,838	253,286,830,121	252,731,683,375	289,632,764	265,513,982	18,611,062,717	1,602,448,954	17,008,613,763

Figure 15 A representative table from the Treasury Bulletin dataset, with nested headers and rows.

Questions in OfficeQA Pro often require retrieving information or performing analysis over data contained in complicated tables. Most tables in the dataset have hierarchical row and column headers (as shown in Figure 15), which are challenging to represent faithfully in text. Prior work has studied the impact of table serialization format on LLM comprehension, finding that format choice significantly affects downstream task performance (Sui et al., 2024). While markdown is a more token-efficient representation than HTML, it doesn’t natively support nested header representations. To address this, we reduce information loss by constructing each markdown header as a collapsed representation of the nested headers. For example, the 2nd column in Figure 15 would be collapsed to a single column header name: “Sept 30, 1990 > Currency > Total”.

Format Comparison: HTML vs. Markdown

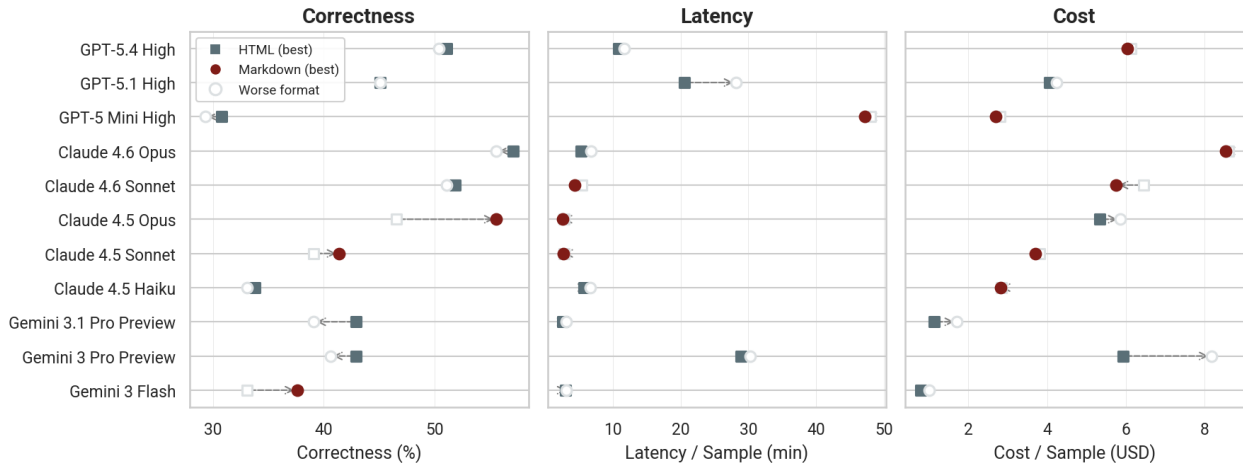


Figure 16 Directional comparison of HTML vs. Markdown table representations across correctness (at 0.0% error threshold), latency, and cost, using the custom agent configured with file search over the full corpus of documents parsed with Databricks’ `ai_parse_document`. Arrows point from HTML to Markdown; filled markers indicate the better format.

In experiments comparing the same base agent using a corpus of tables with HTML vs. Hierarchical Markdown representation, we find that HTML wins overall by a slight margin; improving correctness in 7 out of 11 agents. Results are summarized visually in Figure 16, with full numerical results in Table 5. Performance differences are generally modest, indicating that table serialization format affects performance but does not dominate overall system behavior.

Model-family trends also emerge. All GPT models achieve equal or higher correctness with HTML representations, and both Gemini Pro variants also favor HTML. Claude models exhibit more mixed behavior:

while several variants prefer HTML, others (notably Claude Opus 4.5 and Claude Sonnet 4.5) benefit substantially from Hierarchical Markdown. This variability suggests that sensitivity to table representation is model-dependent and likely reflects differences in training data exposure rather than a universally optimal serialization format.

Across latency, tool usage, and cost, neither representation consistently dominates. Improvements in efficiency metrics are model-specific and typically small relative to overall system variance.

Model	Table Format	Correctness (%)	Latency (min)	Tool Calls	Cost (\$)
GPT-5.4	HTML	51.10	10.87	104.8	\$6.13
	MD	50.40	11.64	114.3	\$6.03
GPT-5.1	HTML	45.10	20.6	60.1	\$4.06
	MD	45.10	28.2	69.1	\$4.25
GPT-5 Mini	HTML	30.80	48.1	120.4	\$2.80
	MD	29.30	47.2	122.7	\$2.69
Claude Opus 4.6	HTML	57.10	5.3	53.3	\$8.61
	MD	55.60	6.7	56.4	\$8.54
Claude Opus 4.5	HTML	46.60	2.8	27.5	\$5.34
	MD	55.60	2.6	29.9	\$5.86
Claude Sonnet 4.6	HTML	51.90	5.4	69.2	\$6.45
	MD	51.10	4.3	60.1	\$5.75
Claude Sonnet 4.5	HTML	39.10	2.8	31.2	\$3.81
	MD	41.40	2.7	29.3	\$3.70
Claude Haiku 4.5	HTML	33.80	5.7	73.9	\$2.84
	MD	33.10	6.6	74.8	\$2.83
Gemini 3.1 Pro Preview	HTML	42.90	2.6	25.7	\$1.13
	MD	39.10	3.1	28.3	\$1.71
Gemini 3 Pro Preview	HTML	42.90	28.9	95.1	\$5.94
	MD	40.60	30.3	102.7	\$8.17
Gemini 3 Flash	HTML	33.10	3.0	40.6	\$0.79
	MD	37.60	3.1	48.6	\$1.01

Table 5 Full performance comparison across HTML and Markdown (MD) table formats using the custom agent with Databricks’ `ai_parse_document` and file search on the full OfficeQA Pro corpus, at 0.0% error threshold.

D.4 Search Tools

Search and retrieval is a critical step in all OfficeQA Pro tasks. In this section, we seek to understand how different search tools compared in terms of their impact on OfficeQA Pro performance. We select one frontier model per provider (GPT-5.4, Claude Opus 4.6, and Gemini 3.1 Pro Preview), each using the best-performing table format from Section D.3 (HTML). We evaluate 3 search configurations: (1) vector search, (2) vector search with contextual embeddings, and (3) file search combined with contextual vector search, comparing against the file search baseline established in the table representation experiments.

Our **file search** experiments use the same tool configuration described in Section 4.1, which allowed for basic file search actions such as `grep`, `sed`, `cat`, and `ls` via sandboxed tools: `fs_search` and `fs_read`.

In **vector search** experiments, the agent has access to a vector database of the corpus. We ingest the pre-parsed documents by concatenating all elements as textual strings. These elements are chunked with overlap (default: 1024 tokens per chunk, with 128 overlap between chunks) using the NLTK Treebank tokenizer. We embed the chunks using `gte-large-en` to produce the vector database. The agent can query this vector index through a custom `vector_search` tool which accepts a query string. Intuitively, in these initial experiments the agent often struggled to find information in chunks that were separated from important semantic information (e.g., a section from the middle of a table, which is disconnected from the table header). This results in an average 27% relative drop in performance compared with file search.

To help make vector search more ergonomic for the agent, we experimented with **vector search with contextual embeddings** (Ford, 2024). Concretely, we appended the following semantic information to each chunk whenever available: the document name, the date and month of the bulletin, the transcribed page number from the page, the page header, and any table names or section titles associated with the page the chunk is on. We find that this improves performance compared with standard vector search by 21% on average, while reducing tool calls by approximately 44%, latency by approximately 38%, and cost by approximately 44%. Compared with file search, contextual vector search is within 12% in performance on average.

Combining file search and vector search with contextual embeddings leads to a further 15% improvement in performance compared with contextual embeddings alone. This combined configuration achieves the highest accuracy in 2 out of 3 models, and for GPT-5.4 and Claude Opus 4.6 is 6–13% cheaper than file search alone.

Interestingly, Gemini 3.1 Pro Preview is the most sensitive to the choice of retrieval method, showing a 39% relative drop from file search to standard vector search. One possible explanation is that, because its overall performance ceiling is lower, the model is more reliant on high-quality retrieval to compensate for limitations in downstream reasoning, tool use, or multi-step synthesis. This suggests that retrieval quality may disproportionately impact weaker model classes.

Model	Search Tool	Correctness (%)	Latency (min)	Tool Calls	Cost (\$)
GPT-5.4	VS	39.10	20.9	220.4	\$12.46
	C-VS	49.60	12.28	123.9	\$7.28
	FS + C-VS	51.90	8.93	86.4	\$5.31
Claude Opus 4.6	VS	45.90	9.04	75.0	\$19.60
	C-VS	49.60	5.16	41.3	\$9.34
	FS + C-VS	53.40	5.16	47.4	\$8.10
Gemini 3.1 Pro Preview	VS	26.30	7.36	56.6	\$5.73
	C-VS	33.80	5.21	33.0	\$3.50
	FS + C-VS	45.10	4.58	26.4	\$1.77

Table 6 Performance comparison at 0.0% error threshold by search tool using the custom agent with access to the full corpus of documents parsed with Databricks’ `ai_parse_document`. VS: vector search with standard embeddings; C-VS: contextual embeddings; FS + C-VS: combined.

D.5 Test-Time Scaling

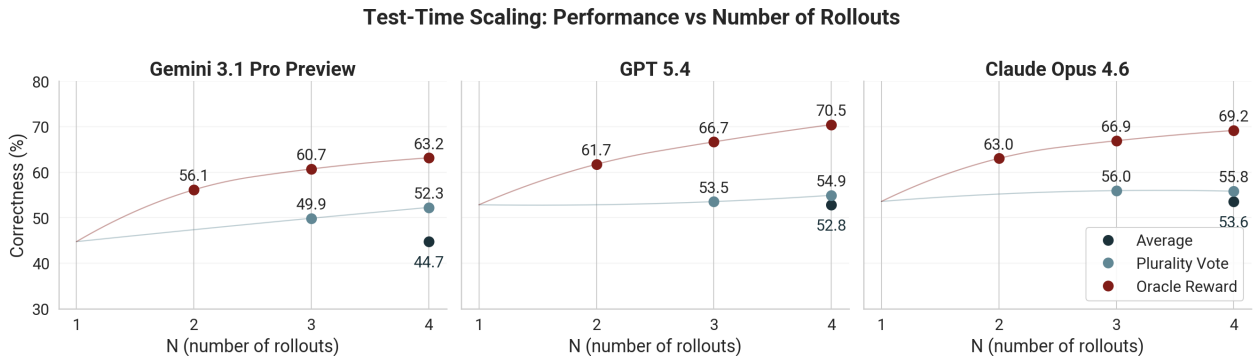


Figure 17 Test-time scaling on OfficeQA Pro using the custom agent with FS + C-VS search over the full corpus parsed with Databricks’ `ai_parse_document`. Correctness is reported at 0.0% error threshold for N=1 to 4 rollouts. Plurality vote and oracle reward strategies improve over single-run average as rollouts increase.

We explore how test time scaling impacts OfficeQA Pro performance. Using the file search + contextual embedding vector search configuration from the previous set of experiments, we run Gemini 3.1 Pro Preview, GPT 5.4, and Claude Opus 4.6 custom agents N=4 times. We experiment with selecting the final answer via a **plurality vote**, where the most common answer from the available N runs is selected as the final answer to be evaluated. If all answers are equally prevalent, we choose an answer at random. We compare this with the **average** performance across runs, as well as with the performance when selecting the final answer from the available N with an **oracle reward** function.

We observe that plurality vote improvements scale inversely with average single-run performance. Agents with lower baseline accuracy show larger gains from aggregation, suggesting higher variance across individual rollouts, whereas stronger agents already produce more consistent answers and therefore benefit less from consensus selection.

E Frontier AI Baseline Experiment Details

E.1 Non-Agent Baseline Results

Model	Config	Correctness (%)
claude-4.6-opus	llm-only	2.26
	llm web-search-only	3.01
	llm-oracle PDF	37.59
	llm-oracle-Databricks parse	50.38
	llm-oracle PDF-web-search	36.09
	llm-oracle-Databricks parse-web-search	57.14
gpt-5.4-high	llm-only	0.75
	llm web-search-only	11.28
	llm-oracle PDF	56.39
	llm-oracle-Databricks parse	62.41
	llm-oracle PDF-web-search	57.14
	llm-oracle-Databricks parse-web-search	65.41
gemini-3.1-pro-preview	llm-only	2.26
	llm web-search-only	3.01
	llm-oracle PDF	51.13
	llm-oracle-Databricks parse	53.38
	llm-oracle PDF-web-search	52.63
	llm-oracle-Databricks parse-web-search	56.39

Table 7 LLM baseline correctness across evaluated configurations.

E.2 Cost Assumptions

Model	Provider	Input (\$/MTok)	Output (\$/MTok)	Cached Input (\$/MTok)
GPT-5.4	OpenAI	2.50	15.00	0.250
GPT-5.1	OpenAI	1.25	10.00	0.125
GPT-5 Mini	OpenAI	0.25	2.00	0.025
Claude Opus 4.6	Anthropic	5.00	25.00	0.50
Claude Opus 4.5	Anthropic	5.00	25.00	0.50
Claude Sonnet 4.6	Anthropic	3.00	15.00	0.30
Claude Sonnet 4.5	Anthropic	3.00	15.00	0.30
Claude Haiku 4.5	Anthropic	1.00	5.00	0.10
Gemini 3.1 Pro Preview	Google	2.00	12.00	—
Gemini 3 Pro Preview	Google	2.00	12.00	—
Gemini 3 Flash	Google	0.50	3.00	—

Table 8 Token pricing assumptions used for cost calculations (USD per 1 million tokens). Sourced from provider pricing pages as of March 2026.

E.3 Preinstalled Libraries for Baseline Agent SDKs

The following packages are preinstalled in the agent sandbox when given PDFs.

System packages (apt): tesseract-ocr, tesseract-ocr-eng, libtesseract-dev, libleptonica-dev, poppler-utils, ghostscript, libgl1, libglib2.0-0, ripgrep, imagemagick, ocrmypdf, qpdf, pdfgrep.

Python packages (pip): pytesseract, easyocr, pymupdf, opencv-python-headless, Pillow, pdf2image, pdfplumber, pypdf, PyPDF2, pdfminer.six, rapidocr-onnxruntime, ocrmypdf, openpyxl, camelot-py[base], tesseract, paddlepaddle, paddleocr, surya-ocr, python-doctr[torch].

This appendix lists the full system prompts used across all baseline configurations. Template variables (e.g. file paths, directory names) are shown with placeholder values.

E.4 Non-Agent (Direct LLM) Prompt

All non-agent baselines share the following base system prompt:

```

1 You are an agent that is an expert in answering questions related to the U.S treasury &
  economy.
2
3 You must always provide an answer that is the result of any computation or reasoning to
  determine the answer to the question.
4 Use maximum precision in your calculations unless otherwise specified by the question.
5
6 REQUIRED FORMAT for completion:
7 When you have the final answer, keep any final reasoning you used before getting to the
  answer and then only return the value in the XML tags.
8
9 <REASONING>
10 [final reasoning - including steps & sources used]
11 </REASONING>
12 <FINAL_ANSWER>
13 [value]
14 </FINAL_ANSWER>
15
16 Never respond with follow-up questions.
17
18 FAILURE CONDITION: If you do not produce a <FINAL_ANSWER> tag, your response will be
  considered incomplete and you will fail the task.

```

Each configuration modifies the base prompt as follows:

- **No Context:** Base system prompt used as-is. The user message contains only the question.
- **Web Search:** The system prompt also includes: “You also have access to web search in the case that you need to look something up.”
- **Oracle PDF:** Base system prompt unchanged. The user message includes the oracle PDF page(s) attached as a base64-encoded image, followed by: “Based on the PDF page(s) shown above, please answer the following question: {question}”
- **Oracle Parsed:** No system message is used. The entire base prompt, the extracted document content, and the question are combined into a single user message.
- **Oracle PDF + Web Search:** Combines the web search addition with the Oracle PDF user message format described above.

E.5 Agent Prompt

The following prompt is used across all three agent baselines (Claude Agent SDK, OpenAI Codex SDK, Gemini CLI) and custom agent setup with file system access and web search enabled.

```

1 You are an agent that is an expert in answering questions related to the U.S treasury &
  economy. Given the question, please use the treasury docs -- these are located in the
  folder ../officeqa_corpus/ (relative to your current working directory), and you can use
  file system search to look through them, given the tools provided. Note that these
  documents are very long, so you should not try to read the whole file. You also have
  access to web search in the case that you need to look something up.
2 If multiple documents report the same metric, use the most up-to-date revision unless the
  question specifies an exact date or document. Do not rely on the first matching value you
  encounter.
3 You must always provide an answer that is the result of any computation or reasoning to
  determine the answer to the question.

```

```
4 Use maximum precision in your calculations unless otherwise specified by the question.
5
6 REQUIRED FORMAT for completion:
7 When you have the final answer, keep any final reasoning you used before getting to the
  answer and then only return the value in the XML tags.
8
9 <REASONING>
10 [final reasoning - including steps & sources used...]
11 </REASONING>
12 <FINAL_ANSWER>
13 [value]
14 </FINAL_ANSWER>
15
16 Never respond with follow-up questions.
17
18 FAILURE CONDITION: If you do not produce a <FINAL_ANSWER> tag, your response will be
  considered incomplete and you will fail the task.
```