DOCOPS: A VERIFIABLE BENCHMARK FOR AUTONOMOUS AGENTS IN COMPLEX DOCUMENT OPERATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

As autonomous agents rapidly evolve, their ability to reliably manipulate ubiquitous digital documents has become critical for enabling general-purpose AI assistants and automating complex workspace workflows. In this paper, we introduce **DocOps**, a deterministically verifiable evaluation framework underpinned by a hierarchical taxonomy that deconstructs document operations inspired by real-world practices into atomic dimensions and escalating workflow complexities. Based on DocOps, we systematically evaluate representative closed- and open-source models across various agentic harnesses, revealing that even the most advanced frontier configurations still exhibit profound limitations when handling highly coupled, long-range tasks. Furthermore, a fine-grained analysis of existing agents' manipulation behaviors uncovers 3 key failure modes: long-term state tracking collapse, shallow semantic verification, and destructive editing of structural metadata. Ultimately, our work exposes the capability boundaries of agents in maintaining global document consistency, shedding light on the future design of robust, non-destructive agents for complex digital ecosystems.

1 INTRODUCTION

Autonomous agents powered by large language models (LLMs) are rapidly evolving from passive conversationalists to active participants in complex digital workspaces (Drouin et al., 2024; Koh et al., 2024; Xie et al., 2024). A central pillar of human productivity involves the creation, modification, and management of ubiquitous digital documents, such as spreadsheets, presentations, word-processed files, and PDFs (Li et al., 2023a; Wang et al., 2024b). As we push toward general-purpose AI assistants, the ability to autonomously navigate and reliably manipulate these highly structured digital artifacts represents a critical frontier in human-computer interaction (Ge et al., 2023; Trivedi et al., 2024).

Despite rapid advancements in agentic capabilities, our quantitative understanding of how models interact with real-world documents remains fundamentally constrained by 2 paradigms (Wang et al., 2024a). On the one hand, static frameworks (e.g., DocBench (Zou et al., 2024), DUDE (Landeghem et al., 2023)) reduce documents to read-only knowledge repositories, benchmarking only information extraction or question-answering over fixed text. On the other end, workflow-oriented evaluations (e.g., OfficeBench (Wang et al., 2024b), OdysseyBench (Wang et al., 2025b)) focus primarily on the orchestration of software navigation, treating the document merely as a transient, passive data payload passed between different applications. Crucially, neither approach treats the document as a first-class computational object, an intricate digital artifact that an agent must actively and continuously manipulate throughout a complex workflow. In real-world productivity scenarios, agents are required to directly synthesize and alter diverse document contents and formats, ensuring that the modified files remain structurally valid and functionally intact. Consequently, it remains an open research question *whether current agents can reliably execute end-to-end document-centric tasks while maintaining global document state consistency, and achieving user objectives without introducing destructive modifications.*

To this end, we introduce **DocOps**, a rigorously verifiable evaluation framework designed to systematically assess LLM agents on complex end-to-end document manipulation. A primary contribution of

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

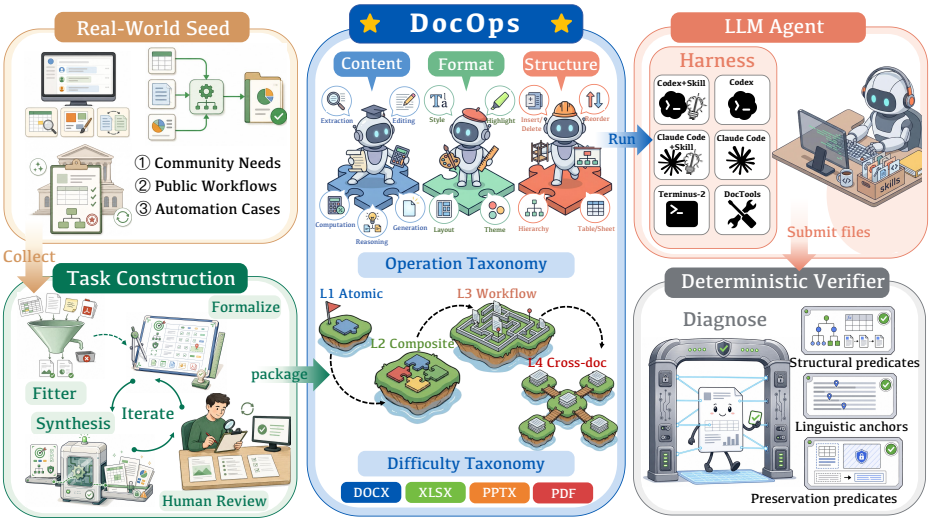


Figure 1: The overview framework of DocOps.

this work is the formulation of a principled taxonomy that systematically deconstructs the operational black box of document manipulation. As illustrated in Figure 1, our taxonomy maps the operational space across two orthogonal axes: atomic capabilities and workflow depth. First, we isolate document interaction into 3 interdependent dimensions: content, format, and structure. Second, we establish a four-tiered complexity gradient. While L1 and L2 isolate single and multi-dimensional atomic edits, L3 and L4 escalate to long-horizons, cross-document workflows inspired by real-world human interaction scenarios. This hierarchical taxonomy serves not merely as a categorization scheme, but as a fine-grained agentic diagnostic framework, enabling researchers to localize whether an agent's failure stems from localized structural disruption or a collapse in long-term state tracking.

Crucially, prior benchmarks often rely on indirect validation targets and coarse task-level success signals, which makes it difficult to fully assess whether agents can produce valid and non-destructively edited office artifacts (Ofengenden et al., 2025; Wang et al., 2024b). Therefore, to ensure evaluation fidelity within this taxonomy, DocOps introduces a deterministic validator for each task that directly targets the native document. Specifically, DocOps directly inspects the output files through native document libraries. Each verifier consists of 3 types of predicates, which can not only check whether the user-requested editing operations have been correctly completed, but also verify that the associated hidden structural invariants are preserved. This deterministic artifact-level design is intended to reduce overly lenient evaluation in document tasks, and further reveals native document failure modes that cannot be captured by relying solely on surface-level validation, task-level success determination, or automated evaluators.

By systematically evaluating a diverse spectrum of leading closed- and open-source models across various agentic harnesses, we uncover the performance limitations and critical bottlenecks of contemporary agent systems. We show that even the most capable frontier configuration (GPT-5.5 (OpenAI, 2026b) with Codex and skills) achieves only an overall success rate of 0.671, and its performance rapidly drops on L3 and L4 tasks, demonstrating that current agent systems still struggle with workflow-level document manipulation. Further analysis reveals that workflow difficulty increases when operations are tightly coupled. For instance, Excel workflows, which demand strict maintenance of formula references and data validation boundaries, see success rates plummet to near zero in complex long-range tasks, whereas tasks in low-coupling formats like PDFs remain relatively robust.

Moreover, we identify 3 pervasive failure modes behind agent failures on complex document tasks: 1) *Long-term state tracking failure*, where agents complete local edits but lose global document state. 2) *Shallow semantic verification*, where agents accept surface-plausible outputs without checking underlying computational or structural semantics. 3) *Destructive editing*, where models persistently downgrade complex object trees into flat text or irreparably break formulas, tables or structural metadata. Our quantitative analysis further shows that these modes account for a majority of failed runs. Finally, we demonstrate that the agent's harness profoundly dictates performance: open-ended programmable coding environments with file-system feedback vastly outperform static RPC-style

tool calling. Furthermore, while explicit procedural guidance (e.g., skills) acts as a strong catalyst for most mid-tier open-source models, it offers marginal utility for frontier models capable of zero-shot orchestration. Together, these findings redefine the trajectory for document-centric AI, shifting the focus from isolated tool invocation to the development of state-aware, non-destructive agents capable of reliable execution within highly coupled digital ecosystems.

Our major contributions are summarized as follows:

- We introduce DocOps, a rigorously verifiable evaluation framework for end-to-end document manipulation.
- We conduct a broad empirical evaluation of leading open- and closed-source models across diverse agentic harnesses, revealing the critical performance bottlenecks of contemporary systems.
- Through fine-grained error analysis, we identify pervasive failure modes in document-centric agent systems.

2 Related Work

Autonomous agents. Autonomous agents have become central to automating complex digital workflows(Guo et al., 2024; Xie et al., 2024). Current research predominantly follows two trajectories. The first focuses on enhancing the intrinsic capabilities of LLMs, developing advanced techniques for tool-use (Li et al., 2023b; Qin et al., 2023), long-context window management (Packer et al., 2024; Hu et al., 2026), and sophisticated memory mechanisms (Wang et al., 2025a; Wei et al., 2025). The second trajectory focuses on harness construction, which provides the scaffolding that encapsulates the execution loop, context manager, and tool registry. Recent work demonstrates that an agent's performance relies heavily on this harness, proving that robust system design is as critical as model intelligence for reliable execution in dynamic environments (Yang et al., 2024; He et al., 2025).

Document-related benchmarks. Existing document benchmarks generally fall into two broad paradigms. The first includes read-only document understanding, such as DocVQA (Mathew et al., 2021), DocBench (Zou et al., 2024), and OmniDocBench (Ouyang et al., 2025), as well as format-specific manipulation, such as SheetCopilot (Li et al., 2023a), SpreadsheetBench (Ma et al., 2024), and PPTArena (Ofengenden et al., 2025). These benchmarks evaluate information extraction or native editing within relatively focused document and task spaces. The second paradigm evaluates longer office workflows and application orchestration, including OfficeBench (Wang et al., 2024b), OdysseyBench (Wang et al., 2025b), DELEGATE-52 (Laban et al., 2026), and FINCH (Dong et al., 2026). OfficeBench and OdysseyBench emphasize application-level workflow completion, DELEGATE-52 measures round-trip degradation under repeated reversible edits, and FINCH evaluates authentic finance workflows with heterogeneous document inputs. Their primary evaluation targets therefore differ from deterministic artifact-level verification that directly checks whether the requested native state is reached while preserving document validity and relevant surrounding state.

A detailed comparison of DocOps with prior benchmarks across task scope, evaluation contract, and agent-system analysis is provided in Appendix A, which *highlights DocOps's central novelty and positions DocOps as a unified framework for evaluating and diagnosing reliable document operations, rather than merely an incremental expansion in formats or task coverage*.

3 DocOps

DocOps is designed to evaluate whether agents can modify native-format office artifacts while preserving their document-native state. It combines a two-axis task taxonomy(Section 3.1), controlled task construction(Section 3.2), and deterministic final-artifact verification(Section 3.3).

3.1 Taxonomy Design

Our taxonomy uses two axes to support both fine-grained operation diagnosis and workflow-level reliability analysis. The **operation axis** targets the distinct editable layers of structured office

documents (Mao et al., 2003). The **difficulty axis** measures an agent's ability to maintain state across complex, multi-step operation sequences (Liu & Li, 2012; Stanton, 2006)

Operation taxonomy. The operation axis categorizes atomic document-editing primitives into three families: 1) *Content* evaluates the model's ability to handle textual or numeric semantics, including extracting information, editing text, generating grounded text, computing formulas and reasoning. 2)*Format* focuses on the presentation state of an artifact and tests whether the model can follow formatting instructions, including style consistency, highlighting key words, layout instructions and transferring theme. 3) *Structure* focuses on the organization of document-native objects across formats, such as reordering slides, inserting a cell, modifying outline levels, and operating on native tables or worksheets.

Difficulty taxonomy. The difficulty axis measures the depth of planning and state-tracking ability required by a task. **L1** contains a single atomic edit and tests local execution. **L2** combines several atomic operations within one artifact and tests short-horizon planning and local dependency modeling. **L3** requires a document-level workflow over one complex artifact and tests whether the agent can maintain global consistency across multiple steps. **L4** requires the production or alignment of a final artifact from multiple documents and tests cross-document state tracking and information alignment.

Task package. Each task is released as a self-contained Harbor bundle (Merrill et al., 2026; Harbor Framework Team, 2026), containing the source artifact(s), a natural-language instruction, optional document skills, and a deterministic verifier. The verifier directly checks the submitted final artifact, allowing the taxonomy to support both agent execution and reproducible evaluation.

3.2 Task Construction

DocOps is a controlled, preservation-aware synthetic benchmark inspired by practical document-operation requests. To balance practical relevance with reproducible evaluation, we construct DocOps through a four-stage pipeline that produces compact, well-specified tasks with explicit editing scopes and preservation requirements.

Stage 1: Seed collection and filtering. We screened 600 candidate sources, including 472 community requests, 125 public workflow references and workflow-level cases, and 3 automation templates. We retained 482 candidates and excluded 118 (19.7%) that required external services or lacked a concrete offline operation, a clear editing scope, verifiable success conditions, redistribution safety, or sufficient distinction from existing patterns. Community requests primarily inform localized operations, while workflow specifications and automation cases inform longer single-document and cross-document workflows. The retained sources form a candidate pool and do not map one-to-one to the final tasks.

Stage 2: Task formalization. We convert each retained seed into a structured specification through standardized prompting and store it in `task_metadata.json`, as detailed in Appendix C. The specification records the taxonomy labels, input and output paths, required operations, editing scope, and preservation requirements. Ambiguous or internally inconsistent specifications are revised before artifact construction.

Stage 3: Source-artifact synthesis. Based on each formal specification, we construct a compact native-format source artifact containing the content, structure, and dependencies required by the task. This includes spreadsheet records and formulas, Word sections, slide objects and PDF pages. The resulting artifacts are placed at the prescribed input paths and retain the task-specific state needed to evaluate scoped editing and preservation.

Stage 4: Iterative human review. One main human reviewer, a PhD-level researcher specializing in document operations, iteratively compares each instruction with its source artifacts and metadata. The review checks instruction clarity, edit scope completeness, taxonomy alignment, native-file integrity, and file path consistency. Tasks that fail these checks are returned to the relevant earlier stage for revision or discarded. For example, an early PowerPoint candidate instructed the agent to modify Slide 2, while the source presentation contained only one slide. Human review detected this mismatch between instructions and artifacts, and the candidate was rejected and corrected before inclusion.

Following this pipeline, DocOps contains **210** tasks, comprising **50** L1 atomic-edit tasks, **40** L2 compositional tasks, **60** L3 single-document workflow tasks, and **60** L4 cross-document workflow

tasks. Each task is packaged as a self-contained Harbor package that contains source artifact(s), a natural language instruction, optional document skills, and a deterministic verifier. Detailed benchmark statistics are provided in Appendix B.

3.3 Verifier Construction and Fidelity

Each task employs a deterministic in-container verifier that inspects the final artifact through document-native libraries. The resulting scoring procedure is deterministic, offline, and independent of LLM-as-a-judge evaluation. To accommodate multiple valid execution paths, the verifier avoids whole-artifact exact matching and instead uses three targeted predicate types. 1) **Structural predicates** validate underlying native states, such as executable formulas and outline hierarchies, to detect structural corruption that may be invisible in rendered outputs. 2) **Linguistic anchors** verify required textual content using task-specific diagnostic keywords, maintaining content accuracy while allowing valid linguistic variation. 3) **Preservation predicates** check whether specified out-of-scope elements, such as protected styles and unmodified worksheets, remain intact.

Verifier fidelity. We directly assess verifier fidelity through a human audit and a mutation-based stress test. For the human audit, a computer science PhD candidate experienced in document operations inspected the task instruction, source artifacts, and submitted artifact for 128 sampled verifier decisions and assigned an artifact-level pass or fail judgment. The verifier agrees with these judgments in 122 cases (**95.31%**), with three false passes and three false fails. Across 180 controlled mutations of 36 verified outputs, it detects **174 violations (96.67%)** spanning requested content, native structure, preservation, missing outputs, and file corruption. Together, these results support verifier fidelity on both sampled outputs and controlled corruptions. Appendix D provides the full protocol, category-level results, and representative disagreement cases.

4 Experimental

4.1 Setup

4.1.1 Models

We evaluate both closed- and open-source model groups, as organized in Table 1. Specifically, we include GPT-5.5 (OpenAI, 2026b), GPT-5.4 (OpenAI, 2026a), and Claude Sonnet 4.6 (Anthropic, 2026). The open-source group includes DeepSeek-V4-Pro (DeepSeek-AI, 2026), Qwen3.6-27B (Qwen Team, 2026b), Qwen3.6-35B-A3B (Qwen Team, 2026c), Qwen3.5-122B-A10B, Qwen3.5-27B, Qwen3.5-35B-A3B, Qwen3.5-9B (Qwen Team, 2026a), Gemma4-31B (Gemma Team, 2026), and GLM-4.5-Air (GLM Team et al., 2025). All evaluated models were publicly accessible at the time of evaluation through provider APIs or publicly available checkpoints. Their exact model IDs, access routes, and serving configurations are provided in Appendix F.

4.1.2 Harnesses

We evaluate four execution harnesses that instantiate different interfaces through which the same model can operate on document tasks. These harnesses cover the main interface regimes from constrained tool use to unrestricted programming. 1) **DocTools:** custom structured document APIs without shell access. 2) **Terminus-2:** a stateful Bash-based harness with file-system access (Harbor Framework, 2026). 3) **Claude Code:** an interactive coding harness with file-system feedback (Anthropic, 2025a). 4) **Codex:** a CLI-based harness for open script execution (OpenAI, 2025).

4.1.3 Skill Setting

For Claude Code and Codex, we evaluate both *w/ skill* and *w/o skill* settings. The skill setting uses the official Anthropic document skills for `xlsx`, `docx`, `pptx`, and `pdf`, released in the Anthropic skills repository (Anthropic, 2025b). In the w/o skill setting, these document-skill modules are removed or made unavailable while keeping the same underlying harness.

Table 1: Overall task pass rates across models and execution harnesses. Bold and underlined entries mark the best and second-best results in each column, respectively. "–" denotes unavailable combinations due to protocol incompatibility, not evaluated failures.

LLM Agents	DocTools	Terminus-2	Codex		Claude Code	
			w/ Skill	w/o Skill	w/ Skill	w/o Skill
Closed-source models						
GPT-5.5	**0.138**	**0.524**	**0.671**	**0.648**	–	–
GPT-5.4	<u>0.119</u>	0.452	<u>0.662</u>	<u>0.638</u>	–	–
Claude Sonnet 4.6	<u>0.119</u>	0.419	–	–	**0.519**	**0.552**
Open-source models						
DeepSeek-V4-Pro	0.057	0.467	0.424	0.400	0.400	<u>0.433</u>
Gemma4-31B	0.033	0.281	0.338	0.324	0.300	0.233
Qwen3.6-27B	0.095	<u>0.476</u>	0.210	0.248	<u>0.424</u>	0.386
Qwen3.6-35B-A3B	0.086	0.386	0.219	0.205	0.338	0.329
Qwen3.5-122B-A10B	0.100	0.362	0.281	0.290	0.290	0.295
Qwen3.5-27B	0.114	0.362	0.295	0.271	0.371	0.300
Qwen3.5-35B-A3B	0.105	0.300	0.176	0.157	0.295	0.262
Qwen3.5-9B	0.081	0.233	0.086	0.062	0.176	0.181
GLM-4.5-Air	0.071	0.162	–	–	0.167	0.129

4.1.4 Evaluation Protocol

All experiments are executed through Harbor (Merrill et al., 2026; Harbor Framework Team, 2026), which manages task containers, agent execution and verifier invocation. Each setting is evaluated on the full 210-task suite. A task is counted as successful only if the agent writes the required output artifact to the designated submission path and the deterministic verifier returns success. Runtime errors, missing outputs, verifier failures and timeouts are counted as failures.

4.1.5 Metrics

We report pass rate as the primary metric:

$$\mathrm{PassRate}(m,h) = \frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}}\{V(m,h,t)=1\},$$

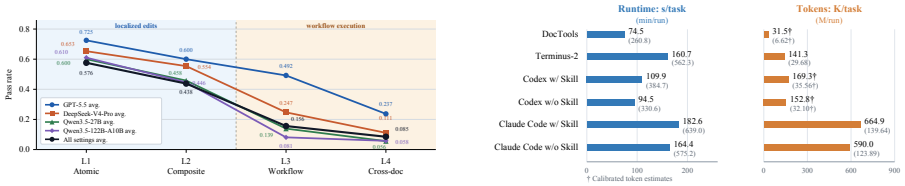
where $V(m,h,t)$ is the verifier result for model m, harness h, and task t, and $\mathcal{T}$ is the 210-task suite. We also report the same metric over subsets grouped by difficulty level, document format, and operation family.

4.2 Results

4.2.1 Main Results

Frontier agents exhibit significant limitations in document manipulation tasks. Table 1 reports the overall pass rates in all the model and harness configurations evaluated. As shown in Table 1, the highest-performing setup (GPT-5.5 with Codex and skills) achieves only a pass rate of 0.671, still failing nearly one-third of the tasks. Furthermore, model capability alone does not dictate success, harness selection is critical. Claude Sonnet 4.6 achieves a competitive pass rate of 0.552 under Claude Code without skills. Terminus-2 also substantially improves several open-source models over direct document-tool use, with Qwen3.6-27B reaching 0.476 under Terminus-2. These results show that document-operation performance depends on the combination of different model capability and execution framework, while even the best current configuration remains far from fully reliable.

Performance collapses once tasks reach workflow-level difficulty. Figure 2(a) shows that most agents perform relatively well on localized and compositional L1/L2 tasks, but degrade sharply once tasks require workflow-level state tracking and cross-step dependency management.For example,



(a) Performance across difficulty levels (b) Harness execution cost

Figure 2: Task difficulty and execution cost. (a) Pass rates across difficulty levels, averaged over available harness and skill settings for each representative model. (b) Mean runtime and token usage per task across harnesses. † denotes calibrated token estimates. Statistical details are in Appendix H.

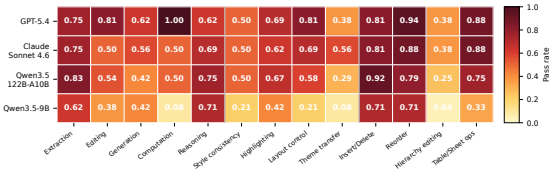
GPT-5.5 decreases from 0.725 on L1 to 0.237 on L4 when averaged over its available harness settings, with similar trends observed for DeepSeek-V4-Pro and the open-source models. Overall, this shared degradation suggests that the central challenge lies not in localized document operations, but in maintaining document state and cross-step dependencies throughout long-horizon workflows.

Richer execution scaffolding tends to carry greater resource overhead, with Claude Code recording the highest cost in both runtime and token usage. To characterize execution efficiency beyond task success, Figure 2(b) reports average runtime and token usage for each harness. DocTools incurs the lowest cost in both dimensions, consistent with its constrained document-tool interface. Compared with Terminus-2, Codex completes tasks substantially faster but consumes moderately more tokens, suggesting that its execution loop exchanges additional model interaction for lower latency. Claude Code exhibits a different profile, recording the highest runtime and token usage across all settings. Enabling skills further increases both costs under Codex and Claude Code. Overall, the execution interface directly shapes the resource profile of a harness, while skill injection introduces additional overhead whose cost must be considered alongside its non-uniform performance effects.

4.2.2 Detailed Analysis

Format	L1		L2		L3
Excel	**0.671** [0.566, 0.773]	-0.238 ⟶	0.433 [0.247, 0.623]	-0.374 ⟶	0.058 [0.031, 0.085]
Word	0.551 [0.382, 0.699]	-0.018 ⟶	**0.533** [0.391, 0.656]	-0.330 ⟶	0.203 [0.140, 0.275]
PPT	0.587 [0.438, 0.715]	-0.187 ⟶	0.400 [0.247, 0.545]	-0.271 ⟶	0.129 [0.070, 0.201]
PDF	0.480 [0.325, 0.641]	-0.095 ⟶	0.386 [0.256, 0.505]	-0.153 ⟶	**0.233** [0.156, 0.316]
FormatCV	0.120 [0.056, 0.279]	+0.011 ⟶	0.131 [0.061, 0.342]	+0.304 ⟶	0.436 [0.321, 0.619]

(a) Performance by document format



(b) Atomic operation accuracy

Figure 3: Performance across document formats and atomic operations. (a) Average pass rates by document format and difficulty level with 95% bootstrap confidence intervals. Arrows indicate changes from the previous difficulty level; FormatCV is computed across formats. (b) Atomic operation accuracy for representative models on L1 tasks, averaged over the available harness settings for each model. The full-model heatmap is provided in Appendix I.

Workflow-level difficulty depends not only on the number of operations, but also on how tightly their underlying document states are coupled. figure 3(a) reports pass rates by document format and difficulty level with 95% bootstrap confidence intervals. We further compute FormatCV to quantify cross-format performance variation:

$$\mathrm{FormatCV}(\ell) = \frac{\mathrm{Std}_{d\in\mathcal{D}}(\mathrm{Acc}_{\ell,d})}{\mathrm{Mean}_{d\in\mathcal{D}}(\mathrm{Acc}_{\ell,d})},$$

where $\mathcal{D} = \{\textsc{Excel}, \textsc{Word}, \textsc{PPT}, \textsc{PDF}\}$. The format-level breakdown shows that workflow degradation is highly uneven. Excel shifts from the highest pass rate at L1 to the lowest at L3 and exhibits the largest decline. This sharp degradation highlights the challenge of jointly maintaining formulas, cell references, sheet order, and hidden states, where a local error can propagate through the workbook. In contrast, PDF retains the highest L3 pass rate, showing that agents remain more reliable when workflow operations involve comparatively modular page-level state changes. Accordingly, FormatCV increases from 0.120 at L1 to 0.436 at L3, with non-overlapping bootstrap confidence

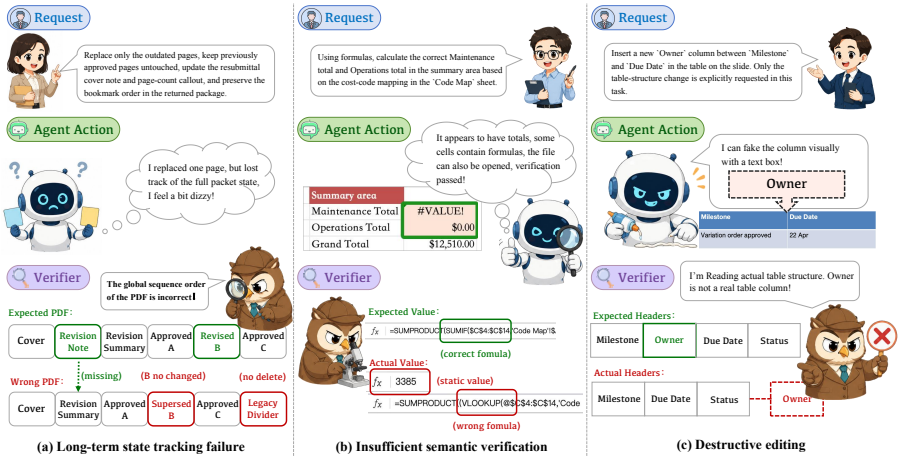


Figure 4: **Representative failure cases** showing typical agent behaviors under three failure modes: (a) long-term state tracking failure. (b) semantic verification gap. (c) destructive editing.

intervals. These results suggest that the coupling of native document states, rather than operation count alone, is a key source of workflow-level difficulty.

Current agents consistently struggle with atomic operations that require document-native state control. Figure 3(b) reports L1 atomic-operation accuracy for representative models. The heatmap shows that operations that rely on localized content access or object movement, such as extraction, reordering, and insert/delete, are relatively reliable. In contrast, hierarchy editing and theme transfer remain difficult, and computation and style consistency exhibit larger variation across model strengths. This suggests that the bottleneck operations for document agents are those that require preserving or updating document-native state, including formulas, styles, themes, and structural hierarchies, rather than simply reading content or moving visible objects.

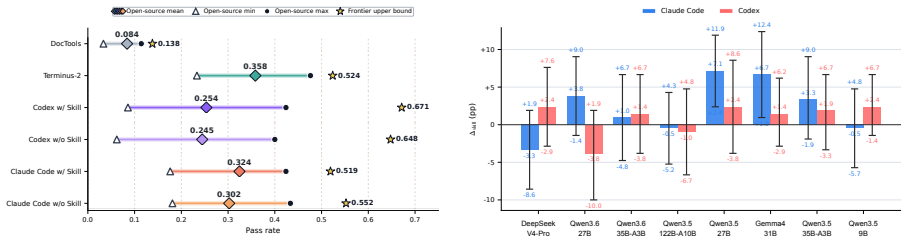
4.2.3 Key Failure Modes

Beyond aggregate pass rates, we analyze triggered verifier assertions and execution trajectories to identify three recurring failure modes: long-term state tracking failure, semantic verification gaps, and destructive editing. Quantitative distributions across representative settings and the overall aggregate are provided in Appendix K. We define three key failure modes as follows.

Long-term state tracking failure: Agents lose global document state during multi-step editing. We define this failure mode as cases where the violated assertions concern persistent document state, such as page, slide, or sheet order, preserved objects, hidden sheets, or cross-document mappings. As shown in Figure 4(a), the agent performs local PDF page edits, but the final document still contains stale or misplaced components. This suggests that the agent fails to maintain a consistent global document plan after each edit, and therefore cannot account for the full document state.

Semantic verification gap: Agents accept visually plausible outputs without verifying executable semantics. We define this failure as cases where an output file is produced but violates task-specific semantic constraints, such as required formulas, value ranges, hierarchy, or structural conditions. In Figure 4(b), the spreadsheet contains plausible static values or seemingly complete formulas, but the key cells contain missing or incorrect formulas. This indicates a clear gap between generic read-back verification and the precise checks required by document-centric workflows.

Destructive editing: Agents complete local edits by damaging native document structure. We define this failure mode as cases where the output breaks document-native invariants, such as formulas, validation rules, styles, heading hierarchy, native tables, bookmarks, or protected reference content. In Figure 4(c), the PowerPoint table appears visually patched, but the underlying native table structure is not correctly updated. Such failures show that agents often treat documents as surface layouts rather than structured editable artifacts.



(a) Performance across harnesses (b) Effect of skill injection

Figure 5: Effects of harnesses and document skills. (a) Pass-rate ranges and means for open-source models with complete harness coverage; stars denote the best. (b) Overall pass-rate change from skill injection. Bars report $\Delta_{\text{skill}} = \text{Acc}_{\text{w/}} - \text{Acc}_{\text{w/o}}$ under Claude Code and Codex.

4.2.4 Impact of Harness and Skills

DocOps shows that the execution harness is a major part of the document agent. The same underlying model can behave very differently depending on whether the harness exposes an open programming environment, a constrained tool interface, or additional procedural skills.

Harness effectiveness depends strongly on the paired model, and open-source configurations remain below their frontier counterparts. Figure 5(a) reports the mean and range of harness-level pass rates across open-source models. The broad ranges show that the same execution framework can yield substantially different outcomes depending on the model. Terminus-2 achieves the strongest open-source aggregate with comparatively stable performance, whereas Codex and Claude Code exhibit greater variation. Thus, greater framework flexibility does not produce uniform gains, since its effectiveness depends on the model's ability to use the execution environment reliably.

The frontier reference markers further illustrate this interaction. Under Codex and Claude Code, the best observed closed-source configurations substantially exceed the corresponding open-source ranges, showing that these harnesses attain much stronger performance when paired with frontier or even official models. Overall, document-operation performance is shaped by the interaction between model capability and execution framework, while current open-source configurations remain considerably behind the strongest evaluated frontier configurations.

Skill injection does not consistently improve document-operation performance and can even constrain effective problem solving. Figure 5(b) reports paired pass-rate differences between skill-enabled and no-skill settings for selected open-source models under Claude Code and Codex. For each model and harness pair, we perform 10,000 paired bootstrap resamples over the common task set and report 95% confidence intervals. Under Claude Code, Qwen3.5-27B and Gemma4-31B improve by 7.1 and 6.7 percentage points, respectively, with intervals entirely above zero. However, DeepSeek-V4-Pro and Qwen3.5-9B show negative point estimates, and several intervals overlap zero. Under Codex, the effects are generally smaller and less certain: Qwen3.6-27B decreases by 3.8 points, while most other intervals cross zero. These results show that the skill package injection does not consistently improve document-operation performance and should be treated as an empirical design choice. Fine-grained trajectory analysis further suggests that overly rigid adherence to prescribed procedures can limit adaptation and alternative problem-solving strategies across diverse tasks. Appendix J provides the full breakdown by difficulty level and operation family.

Conclusion

In conclusion, DocOps establishes a controlled, preservation-aware benchmark for evaluating autonomous agents on native-format artifacts. Across the evaluated model–harness configurations, performance declines substantially as tasks require longer-horizon state tracking, cross-step dependency management, and preservation of document-native structure. Our analysis identifies recurring failures in maintaining global state, verifying task-specific semantics, and preserving native structure. Ultimately, DocOps provides a reproducible diagnostic framework and empirical foundation for guiding the development of next-generation agents capable of maintaining global consistency and avoiding destructive modifications across complex digital artifacts. We will expand the continue expanding the suite by adding more document domains and larger workflow packet while maintaining deterministic verification and human quality control in the future.

AI USE STATEMENT

We used generative AI tools to assist with writing and polish the manuscript for clarity and readability. We also used LLMs in the benchmark data synthesis pipeline, as detailed in Section 3.2. The authors reviewed and revised the AI-assisted text and manually reviewed the generated task specifications and source artifacts. We take full responsibility for the final manuscript and benchmark.

REFERENCES

Anthropic. Claude code: an agentic coding tool, 2025a. URL https://github.com/anthropics/claude-code.

Anthropic. Anthropic skills. https://github.com/anthropics/skills, 2025b.

Anthropic. Introducing claude sonnet 4.6. https://www.anthropic.com/news/claude-sonnet-4-6, 2026.

DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026.

Haoyu Dong, Pengkun Zhang, Yan Gao, Xuanyu Dong, Yilin Cheng, Mingzhe Lu, Zikun Zhu, Adina Yakefu, and Shuxin Zheng. Finch: Benchmarking finance & accounting across spreadsheet-centric enterprise workflows, 2026. URL https://arxiv.org/abs/2512.13168.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. Workarena: how capable are web agents at solving common knowledge work tasks? In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Yingqiang Ge, Yujie Ren, Wenyue Hua, Shuyuan Xu, Juntao Tan, and Yongfeng Zhang. Llm as os, agents as apps: Envisioning aios, agents and the aios-agent ecosystem, 2023. URL https://arxiv.org/abs/2312.03815.

Gemma Team. Gemma 4 technical report, 2026. URL https://arxiv.org/abs/2607.02770.

GLM Team, Aohan Zeng, Xin Lv, Qinkai Zheng, Zhenyu Hou, Bin Chen, Chengxing Xie, Cunxiang Wang, Da Yin, Hao Zeng, Jiajie Zhang, Kedong Wang, Lucen Zhong, Mingdao Liu, Rui Lu, Shulin Cao, Xiaohan Zhang, Xuancheng Huang, Yao Wei, Yean Cheng, Yifan An, Yilin Niu, Yuanhao Wen, Yushi Bai, Zhengxiao Du, Zihan Wang, Zilin Zhu, Bohan Zhang, Bosi Wen, Bowen Wu, Bowen Xu, Can Huang, Casey Zhao, Changpeng Cai, Chao Yu, Chen Li, Chendi Ge, Chenghua Huang, Chenhui Zhang, Chenxi Xu, Chenzheng Zhu, Chuang Li, Congfeng Yin, Daoyan Lin, Dayong Yang, Dazhi Jiang, Ding Ai, Erle Zhu, Fei Wang, Gengzheng Pan, Guo Wang, Hailong Sun, Haitao Li, Haiyang Li, Haiyi Hu, Hanyu Zhang, Hao Peng, Hao Tai, Haoke Zhang, Haoran Wang, Haoyu Yang, He Liu, He Zhao, Hongwei Liu, Hongxi Yan, Huan Liu, Huilong Chen, Ji Li, Jiajing Zhao, Jiamin Ren, Jian Jiao, Jiani Zhao, Jianyang Yan, Jiaqi Wang, Jiayi Gui, Jiayue Zhao, Jie Liu, Jijie Li, Jing Li, Jing Lu, Jingsen Wang, Jingwei Yuan, Jingxuan Li, Jingzhao Du, Jinhua Du, Jinxin Liu, Junkai Zhi, Junli Gao, Ke Wang, Lekang Yang, Liang Xu, Lin Fan, Lindong Wu, Lintao Ding, Lu Wang, Man Zhang, Minghao Li, Minghuan Xu, Mingming Zhao, Mingshu Zhai, Pengfan Du, Qian Dong, Shangde Lei, Shangqing Tu, Shangtong Yang, Shaoyou Lu, Shijie Li, Shuang Li, Shuang-Li, Shuxun Yang, Sibo Yi, Tianshu Yu, Wei Tian, Weihan Wang, Wenbo Yu, Weng Lam Tam, Wenjie Liang, Wentao Liu, Xiao Wang, Xiaohan Jia, Xiaotao Gu, Xiaoying Ling, Xin Wang, Xing Fan, Xingru Pan, Xinyuan Zhang, Xinze Zhang, Xiuqing Fu, Xunkai Zhang, Yabo Xu, Yandong Wu, Yida Lu, Yidong Wang, Yilin Zhou, Yiming Pan, Ying Zhang, Yingli Wang, Yingru Li, Yinpei Su, Yipeng Geng, Yitong Zhu, Yongkun Yang, Yuhang Li, Yuhao Wu, Yujiang Li, Yunan Liu, Yunqing Wang, Yuntao Li, Yuxuan Zhang, Zezhen Liu, Zhen Yang, Zhengda Zhou, Zhongpei Qiao, Zhuoer Feng, Zhuorui Liu, Zichen Zhang, Zihan Wang, Zijun Yao, Zikang Wang, Ziqiang Liu, Ziwei Chai, Zixuan Li, Zuodong Zhao, Wenguang Chen, Jidong Zhai, Bin Xu, Minlie Huang, Hongning Wang, Juanzi Li, Yuxiao Dong, and Jie Tang. Glm-4.5: Agentic, reasoning, and coding (arc) foundation models, 2025. URL https://arxiv.org/abs/2508.06471.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. Large language model based multi-agents: A survey of progress and

challenges. In Kate Larson (ed.), *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pp. 8048–8057. International Joint Conferences on Artificial Intelligence Organization, 8 2024. doi: 10.24963/ijcai.2024/890. URL https://doi.org/10.24963/ijcai.2024/890. Survey Track.

Harbor Framework. Terminus-2: Harbor's reference agent implementation. https://harborframework.com/docs/agents/terminus-2, 2026.

Harbor Framework Team. Harbor: A framework for evaluating and optimizing agents and models in container environments, January 2026. URL https://github.com/harbor-framework/harbor.

Junda He, Christoph Treude, and David Lo. Llm-based multi-agent systems for software engineering: Literature review, vision, and the road ahead. *ACM Trans. Softw. Eng. Methodol.*, 34(5), May 2025. ISSN 1049-331X. doi: 10.1145/3712003. URL https://doi.org/10.1145/3712003.

Yuyang Hu, Shichun Liu, Yanwei Yue, Guibin Zhang, Boyang Liu, Fangyi Zhu, Jiahang Lin, Honglin Guo, Shihan Dou, Zhiheng Xi, Senjie Jin, Jiejun Tan, Yanbin Yin, Jiongnan Liu, Zeyu Zhang, Zhongxiang Sun, Yutao Zhu, Hao Sun, Boci Peng, Zhenrong Cheng, Xuanbo Fan, Jiaxin Guo, Xinlei Yu, Zhenhong Zhou, Zewen Hu, Jiahao Huo, Junhao Wang, Yuwei Niu, Yu Wang, Zhenfei Yin, Xiaobin Hu, Yue Liao, Qiankun Li, Kun Wang, Wangchunshu Zhou, Yixin Liu, Dawei Cheng, Qi Zhang, Tao Gui, Shirui Pan, Yan Zhang, Philip Torr, Zhicheng Dou, Ji-Rong Wen, Xuanjing Huang, Yu-Gang Jiang, and Shuicheng Yan. Memory in the age of ai agents, 2026. URL https://arxiv.org/abs/2512.13564.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Russ Salakhutdinov, and Daniel Fried. VisualWebArena: Evaluating multimodal agents on realistic visual web tasks. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 881–905, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.50. URL https://aclanthology.org/2024.acl-long.50/.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Philippe Laban, Tobias Schnabel, and Jennifer Neville. Llms corrupt your documents when you delegate, 2026. URL https://arxiv.org/abs/2604.15597.

Jordy Van Landeghem, Rafal Powalski, Rubèn Tito, Dawid Jurkiewicz, Matthew B. Blaschko, Lukasz Borchmann, Mickaël Coustaty, Sien Moens, Michal Pietruszka, Bertrand Anckaert, Tomasz Stanislawek, Pawel Józiak, and Ernest Valveny. Document understanding dataset and evaluation (DUDE). In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pp. 19471–19483. IEEE, 2023. doi: 10.1109/ICCV51070.2023.01789. URL https://doi.org/10.1109/ICCV51070.2023.01789.

Hongxin Li, Jingran Su, Yuntao Chen, Qing Li, and ZHAO-XIANG ZHANG. Sheetcopilot: Bringing software productivity to the next level through large language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 4952–4984. Curran Associates, Inc., 2023a. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/0ff30c4bf31db0119a6219e0d250e037-Paper-Conference.pdf.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. API-bank: A comprehensive benchmark for tool-augmented LLMs. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3102–3116, Singapore, December 2023b. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.187. URL https://aclanthology.org/2023.emnlp-main.187/.

Peng Liu and Zhizhong Li. Task complexity: A review and conceptualization framework. *International Journal of Industrial Ergonomics*, 42(6):553–568, 2012. ISSN 0169-8141. doi: https://doi.org/10.1016/j.ergon.2012.09.001. URL https://www.sciencedirect.com/science/article/pii/S0169814112000868.

Zeyao Ma, Bohan Zhang, Jing Zhang, Jifan Yu, Xiaokang Zhang, Xiaohan Zhang, Sijia Luo, Xi Wang, and Jie Tang. Spreadsheetbench: Towards challenging real world spreadsheet manipulation. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 94871–94908. Curran Associates, Inc., 2024. doi: 10.52202/079017-3007. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/ac840df270ac537dd74530a15c332684-Paper-Datasets_and_Benchmarks_Track.pdf.

Song Mao, Azriel Rosenfeld, and Tapas Kanungo. Document structure analysis algorithms: a literature survey. In Tapas Kanungo, Elisa H. Barney Smith, Jianying Hu, and Paul B. Kantor (eds.), *Document Recognition and Retrieval X*, volume 5010, pp. 197 – 207. International Society for Optics and Photonics, SPIE, 2003. doi: 10.1117/12.476326. URL https://doi.org/10.1117/12.476326.

Minesh Mathew, Dimosthenis Karatzas, and C.V. Jawahar. Docvqa: A dataset for vqa on document images. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 2200–2209, January 2021.

Mike A. Merrill, Alexander G. Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E. Kelly Buchanan, Junhong Shen, Guanghao Ye, Haowei Lin, Jason Poulos, Maoyu Wang, Marianna Nezhurina, Jenia Jitsev, Di Lu, Orfeas Menis Mastromichalakis, Zhiwei Xu, Zizhao Chen, Yue Liu, Robert Zhang, Leon Liangyu Chen, Anurag Kashyap, Jan-Lucas Uslu, Jeffrey Li, Jianbo Wu, Minghao Yan, Song Bian, Vedang Sharma, Ke Sun, Steven Dillmann, Akshay Anand, Andrew Lanpouthakoun, Bardia Koopah, Changran Hu, Etash Guha, Gabriel H. S. Dreiman, Jiacheng Zhu, Karl Krauth, Li Zhong, Niklas Muennighoff, Robert Amanfu, Shangyin Tan, Shreyas Pimpalgaonkar, Tushar Aggarwal, Xiangning Lin, Xin Lan, Xuandong Zhao, Yiqing Liang, Yuanli Wang, Zilong Wang, Changzhi Zhou, David Heineman, Hange Liu, Harsh Trivedi, John Yang, Junhong Lin, Manish Shetty, Michael Yang, Nabil Omi, Negin Raoof, Shanda Li, Terry Yue Zhuo, Wuwei Lin, Yiwei Dai, Yuxin Wang, Wenhao Chai, Shang Zhou, Dariush Wahdany, Ziyu She, Jiaming Hu, Zhikang Dong, Yuxuan Zhu, Sasha Cui, Ahson Saiyed, Arinbjörn Kolbeinsson, Jesse Hu, Christopher Michael Rytting, Ryan Marten, Yixin Wang, Alex Dimakis, Andy Konwinski, and Ludwig Schmidt. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces, 2026. URL https://arxiv.org/abs/2601.11868.

Michael Ofengenden, Yunze Man, Ziqi Pang, and Yu-Xiong Wang. Pptarena: A benchmark for agentic powerpoint editing, 2025. URL https://arxiv.org/abs/2512.03042.

OpenAI. Openai codex cli. https://help.openai.com/en/articles/11096431-openai-codex-cli-getting-started, 2025. Accessed 2026-05-21.

OpenAI. Introducing gpt-5.4. https://openai.com/index/introducing-gpt-5-4/, 2026a.

OpenAI. Introducing gpt-5.5. https://openai.com/index/introducing-gpt-5-5/, 2026b.

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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24838–24848, June 2025.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems, 2024. URL https://arxiv.org/abs/2310.08560.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein,

Dahai Li, Zhiyuan Liu, and Maosong Sun. Toolllm: Facilitating large language models to master 16000+ real-world apis, 2023. URL https://arxiv.org/abs/2307.16789.

Qwen Team. Qwen3.5: Accelerating productivity with native multimodal agents, February 2026a. URL https://qwen.ai/blog?id=qwen3.5.

Qwen Team. Qwen3.6-27B: Flagship-level coding in a 27b dense model, April 2026b. URL https://qwen.ai/blog?id=qwen3.6-27b.

Qwen Team. Qwen3.6-35B-A3B: Agentic coding power, now open to all, April 2026c. URL https://qwen.ai/blog?id=qwen3.6-35b-a3b.

Neville A. Stanton. Hierarchical task analysis: Developments, applications, and extensions. *Applied Ergonomics*, 37(1):55–79, 2006. ISSN 0003-6870. doi: https://doi.org/10.1016/j.apergo.2005.06.003. URL https://www.sciencedirect.com/science/article/pii/S0003687005000980. Special Issue: Fundamental Reviews.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16022–16076, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.850. URL https://aclanthology.org/2024.acl-long.850/.

Bo Wang, Weiyi He, Shenglai Zeng, Zhen Xiang, Yue Xing, Jiliang Tang, and Pengfei He. Unveiling privacy risks in LLM agent memory. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 25241–25260, Vienna, Austria, July 2025a. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1227. URL https://aclanthology.org/2025.acl-long.1227/.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6), March 2024a. ISSN 2095-2236. doi: 10.1007/s11704-024-40231-1. URL http://dx.doi.org/10.1007/s11704-024-40231-1.

Weixuan Wang, Dongge Han, Daniel Madrigal Diaz, Jin Xu, Victor Rühle, and Saravan Rajmohan. Odysseybench: Evaluating llm agents on long-horizon complex office application workflows, 2025b. URL https://arxiv.org/abs/2508.09124.

Zilong Wang, Yuedong Cui, Li Zhong, Zimin Zhang, Da Yin, Bill Yuchen Lin, and Jingbo Shang. Officebench: Benchmarking language agents across multiple applications for office automation, 2024b. URL https://arxiv.org/abs/2407.19056.

Qianshan Wei, Tengchao Yang, Yaochen Wang, Xinfeng Li, Lijun Li, Zhenfei Yin, Yi Zhan, Thorsten Holz, Zhiqiang Lin, and XiaoFeng Wang. A-memguard: A proactive defense framework for llm-based agent memory, 2025. URL https://arxiv.org/abs/2510.02373.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 52040–52094. Curran Associates, Inc., 2024. doi: 10.52202/079017-1650. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/5d413e48f84dc61244b6be550f1cd8f5-Paper-Datasets_and_Benchmarks_Track.pdf.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering, 2024. URL https://arxiv.org/abs/2405.15793.

Anni Zou, Wenhao Yu, Hongming Zhang, Kaixin Ma, Deng Cai, Zhuosheng Zhang, Hai Zhao, and Dong Yu. Docbench: A benchmark for evaluating llm-based document reading systems, 2024. URL https://arxiv.org/abs/2407.10701.

APPENDIX

A COMPARISON WITH RELATED BENCHMARKS

Table 2 compares DocOps with representative document-understanding, document-editing, and office-workflow benchmarks across task scope, evaluation contract, and agent-system analysis. The comparison includes read-only document understanding, format-specific editing, application-level workflows, round-trip document preservation, and spreadsheet-centric finance workflows. These benchmarks address complementary research questions; the comparison characterizes their evaluation priorities rather than providing an overall ranking.

Table 2: Comparison with representative document benchmarks across supported formats and explicit evaluation or analysis dimensions.

Benchmark	Formats	Native file edit	Cross-doc workflow	Deterministic target verifier	Explicit preservation	Harness/skill analysis
DocBench (Zou et al., 2024)	PDF	×	×	×	×	×
SpreadsheetBench (Ma et al., 2024)	XLSX	✓	×	×	×	×
PPTArena (Ofengenden et al., 2025)	PPTX	✓	×	×	×	×
OfficeBench (Wang et al., 2024b)	DOCX, XLSX, PDF + apps	✓	✓	✓	×	×
OdysseyBench (Wang et al., 2025b)	DOCX, XLSX, PDF + apps	✓	✓	✓	×	×
DELEGATE-52 (Laban et al., 2026)	Markdown, CSV, TXT, TeX, etc.	✓	✓	×	✓	×
FINCH (Dong et al., 2026)	XLSX + PDF/DOCX inputs	✓	✓	×	×	×
DocOps (ours)	**XLSX, DOCX, PPTX, PDF**	✓	✓	✓	✓	✓

In addition to supported file formats, we consider five comparison dimensions below:

• **Native-file editing.** Tasks require modifying native document artifacts, rather than only answering questions about their contents.

• **Cross-document workflows.** Tasks require coordinating information or operations across multiple documents to produce the requested outputs.

• **Deterministic target verification.** Evaluation directly checks the requested final document state through programmatic predicates. This criterion distinguishes target-state verification from round-trip reconstruction tests and LLM-as-a-judge evaluation; it does not imply exhaustive verification of every document property.

• **Explicit preservation.** Evaluation explicitly checks whether designated content, structure, or other document properties remain intact. Preservation may be assessed through different protocols, including round-trip tests or task-specific checks of out-of-scope state.

• **Harness/skill analysis.** The benchmark systematically studies how execution harnesses or procedural skills affect agent performance, rather than reporting results under a single execution setup.

Interpretation of the comparison. A checkmark indicates that a property constitutes an explicit benchmark-level evaluation target or analysis dimension. A cross indicates that the property is not explicitly established under this comparison criterion; it should not be interpreted as evidence that the benchmark contains no related tasks, tools, or checks. Likewise, support for a document format does not by itself establish that its native structure or preservation requirements are evaluated. For example, reading a PDF as an input differs from editing and verifying its native document state. We therefore distinguish task coverage from the evaluation contract applied to the resulting artifacts.

Format	L1	L2	L3	L4
Excel	14	10	15	84
Word	12	10	15	115
PowerPoint	12	10	15	13
PDF	12	10	15	101

(a) Source artifacts by format and difficulty

Content	Total: 354
C1 Extraction	78
C2 Editing	94
C3 Generation	57
C4 Computation	66
C5 Reasoning	59

Format	Total: 165
F1 Style consistency	66
F2 Highlighting	47
F3 Layout control	26
F4 Theme transfer	26

Structure	Total: 270
S1 Insert/Delete	51
S2 Reorder	69
S3 Hierarchy editing	63
S4 Table/Sheet operations	87

(b) Operation-label counts by family (210 tasks)

Figure 6: DocOps benchmark statistics.

DocOps's central contribution lies in *combining heterogeneous document operations with explicit target-state and preservation checks. It treats documents as first-class, stateful computational objects: each operation is evaluated as a state transition that must reach the requested state while preserving native-format validity and specified out-of-scope document state.* The task taxonomy further supports analysis across atomic operations and increasingly complex workflows, while the model, harness and skill comparisons expose the role of execution scaffolding in document-operation reliability. Together, these design choices position DocOps as a unified framework for evaluating and diagnosing reliable document operations, beyond an expansion in formats or task coverage alone.

B Task Statistics and Full Task List

Figure 6 presents two complementary views of the expanded 210-task DocOps benchmark. Panel (a) reports the number of source artifacts by format and difficulty level, rather than the number of tasks. This distinction is particularly important for L4 cross-document tasks, since a single task may contain multiple source artifacts in different formats. Panel (b) shows the distribution of operation labels across the full benchmark under the Content, Format, and Structure taxonomy. Because composite and workflow-level tasks may involve multiple primitives, these values represent label occurrences rather than mutually exclusive task counts. Accordingly, each family total is the sum of its primitive-label occurrences.

Table 6 provides the complete listing of all 210 tasks and is placed at the end of the appendix for readability. For each task, we report the task identifier, difficulty level, operation labels, normalized input formats, and required output format. For L4 cross-document tasks, the input-format column includes all unique source formats contained in the task environment.

Task Formalization Prompt. Given an informal seed describing a document-editing need, convert it into a structured `task_metadata.json` specification for a self-contained benchmark task.
Input: `{SEED_TEXT}`
Output: Return only valid JSON using the fields below.
Common fields for all tasks: `task_id`; `task_dir_name`; `doc_type`; `primary_modality`; `difficulty`; `verifier_mode`; `user_task_description`; `single_step_benchmark_instruction`; `atomic_scope`; `expected_output_type`; `notes`; `input_path`; `output_path`; `output_filename`; `output_extension`; `output_kind`.
Fields for L1 atomic tasks: `atomic_operation`; `source_doc_filename`; `source_doc_path`; `task_description_filename`; `task_description_path`; `inspired_by`.
Fields for composite and workflow tasks: `atomic_operations`; `num_atomic_operations`; `composition_type`; `composition_pattern`; `source_type`.
Additional field for cross-document tasks: `supporting_input_paths`.
Optional field when the final state is fixed by the task design: `verifier_expectations`.
Constraints: The task must require editing a real document artifact; be solvable offline inside a self-contained container; have a clear required output path; preserve out-of- scope content specified in `atomic_scope`; and avoid subjective requests such as purely aesthetic improvement. Use `notes` to record latent issues or task-specific preservation requirements that should be reflected in the source artifact.

Figure 7: Prompt template used to formalize informal document-editing seeds into the metadata schema used by DocOps.

C Task Formalization Prompt

Figure 7 shows the prompt template used in Stage 2 of the construction pipeline. The goal of this prompt is not to generate the final document artifact directly, but to convert an informal document-editing seed into a structured metadata specification. This metadata acts as the task-level contract

for later artifact synthesis and review: it fixes the document format, difficulty level, operation labels, input and output paths, user-facing instruction, editing scope, expected output type, and task-specific notes. For composite, workflow, and cross-document tasks, the same schema additionally records the operation sequence and supporting input files.

D VERIFIER FIDELITY EVALUATION

We evaluate verifier fidelity through two complementary experiments. The manual audit measures agreement on actual agent outputs, while the mutation stress test examines whether controlled document corruptions are detected.

Manual artifact-level audit. We sampled 128 verifier decisions spanning multiple document formats and difficulty levels. A computer science PhD candidate experienced in document operations inspected the instruction, source artifacts, and submitted output for each case. The auditor assigned a pass or fail judgment based on completion of the requested edits, correctness of native structure, preservation of relevant out-of-scope state, and validity of the required output. We then compared these judgments with the deterministic verifier decisions. As shown in Table 3, the verifier agrees with the manual judgments in 122 of 128 cases (95.31%), with three false passes and three false fails.

One representative false pass correctly reordered the requested slides but also altered the presentation's global canvas dimensions, which were not covered by the verifier. One false fail correctly removed an initial blank page but was rejected by an over-specific predicate tied to a particular low-level page-break representation. We consider these cases acceptable residual edge cases for deterministic artifact validation, as they involve peripheral document properties or implementation-level representations and occur infrequently enough to avoid materially affecting the benchmark conclusions.

Mutation-based stress test. We selected 36 outputs accepted by both the manual audit and the verifier, then introduced five controlled mutations into each output. These mutations altered requested content, corrupted native structure, violated preservation requirements, deleted the required output, or damaged the file while retaining its extension. The verifier detects 174 of the 180 mutations (96.67%), including all missing-output and parse-corruption cases.

To further analyze the undetected mutation cases, we conducted a case study for error analysis. One representative acceptable mistake is an Excel editing task where the required operation was a localized rewrite, while the mutation only affected a worksheet-view detail unrelated to the task. Since all task-related content and the workbook itself remained valid, and the required operation did not involve that worksheet view totally, we consider this case acceptable under the task scope. Overall, these results support the reliability of the verifier suite, while the few missed cases help quantify concrete boundary cases.

Table 3: Verifier-fidelity results. False passes and false fails are defined relative to manual artifact-level judgments. Mutation rates report the proportion of injected violations detected by the verifier.

Evaluation outcome	**Result**
Manual audit (128 decisions)	
Agreement	122/128 (95.31%)
False pass	3
False fail	3
Mutation stress test (36 tasks)	
Requested-content error	35/36 (97.22%)
Native-structure error	33/36 (91.67%)
Preservation error	34/36 (94.44%)
Required output deleted	36/36 (100.00%)
Same-extension parse corruption	36/36 (100.00%)
Overall	**174/180 (96.67%)**

E REPRESENTATIVE VERIFIER SNIPPETS

Each task verifier is written as a deterministic post-condition checker over the submitted artifact. The goal is not to enforce a single edit script, but to check whether the final document satisfies the

semantic and structural contract of the task. Figure 8 to Figure 11 show representative snippets from four formats.

Excel formula verification. Spreadsheet tasks often admit visually plausible but semantically wrong outputs: a model can type the correct-looking number while failing to preserve executable spreadsheet logic. The Excel verifier therefore checks both *formula presence* and *formula semantics*. It accepts several valid formula families, but rejects static values and formulas that are not grounded in the required source ranges.

```
def verify_task():
  ws = load_xlsx_pair(OUTPUT_PATH)[0]["Cost Tracker"]
  assert ws["I3"].value.startswith("=")
  assert ws["I4"].value.startswith("=")
  formula = normalize_text(ws["I3"].value)
  assert "sumif" in formula or "sumproduct" in formula
  mapping = code_map(wb["Code Map"])
  assert_formula_total(ws, mapping, "I3", "Maintenance", 9125.00)
  assert_formula_total(ws, mapping, "I4", "Operations", 3385.00)
```

Figure 8: Representative Excel verifier snippet. The check distinguishes executable formulas from static numeric answers and recomputes expected totals from the workbook state.

Word hierarchy and table verification. Word tasks frequently involve hidden document structure: heading styles, section boundaries, and native tables are not equivalent to plain visible text. The Word verifier reads the `.docx` object model, extracts styled heading paragraphs, and inspects native table cells directly. This prevents an agent from passing by only drawing text that looks like a heading or table.

```
def verify_task():
  doc = Document(OUTPUT_PATH)
  headings = heading_paragraphs(doc)
  assert headings == EXPECT["heading_sequence"]
  for spec in EXPECT["paragraphs_under_headings"]:
    actual = paragraphs_under_heading(doc, spec["heading"])
    assert actual == spec["exact_lines"]
  table = doc.tables[EXPECT["table"]["index"]]
  assert table.cell(0, 0).text.strip() == "Item"
  assert len(table.columns) == EXPECT["table"]["cols"]
```

Figure 9: Representative Word verifier snippet. The verifier grades heading hierarchy and native table state through the Word object model.

PowerPoint native table verification. PowerPoint is especially prone to destructive visual patches: an agent may overlay text boxes that resemble a table while leaving the underlying slide table unchanged. The verifier therefore searches for a real `shape.has_table` object and validates the native table grid and preserved row content.

```
def verify_task():
  prs = Presentation(OUTPUT_PATH)
  slide = prs.slides[0]
  table_shape = next(s for s in slide.shapes if s.has_table)
  table = table_shape.table
  headers = [table.cell(0, c).text.strip()
    for c in range(len(table.columns))]
  assert headers == ["Milestone", "Owner", "Due Date", "Status"]
  row = [table.cell(1, c).text.strip() for c in range(len(table.columns))]
  assert row == ["Variation order approved", "", "22 Apr", "Open"]
```

Figure 10: Representative PowerPoint verifier snippet. The check requires a native PowerPoint table rather than visually similar text boxes.

PDF page-order and outline verification. PDF tasks combine visual page content with document navigation state. The PDF verifier therefore checks the visible page order and the outline tree together: page text confirms that the physical sequence is correct, while extracted outline titles and levels confirm that bookmarks were rebuilt rather than merely ignored.

```
def verify_task():
  pages = pdf_page_texts(OUTPUT_PATH)
  assert normalize_text(pages[0]).startswith("scope")
  assert normalize_text(pages[1]).startswith("findings")
  title = normalize_text(pages[2])
  assert title.startswith("recommendations")
  outline = extract_pdf_outline_titles(OUTPUT_PATH)
  expected = [(0,"Scope"), (0,"Findings"), (1,"Site A"),
    (1,"Site B"), (0,"Recommendations"), (0,"Appendix")]
  assert outline == expected
```

Figure 11: Representative PDF verifier snippet. The verifier jointly checks page order and bookmark hierarchy.

F Model Serving Details

All evaluated models were publicly accessible at the time of evaluation. Their access and serving configurations are summarized as follows:

Table 4: Model serving configurations for locally evaluated open-weight models.

Model	Served name	Max model lens	TP	Parser
Qwen3.6-27B	`local-qwen36-27b`	131072	8	`qwen3_coder`
Qwen3.6-35B-A3B	`local-qwen36-35b`	131072	8	`qwen3_coder`
Qwen3.5-122B-A10B	`local-qwen35-122b`	262144	8	`qwen3_coder`
Qwen3.5-27B	`local-qwen35-27b`	131072	8	`qwen3_coder`
Qwen3.5-35B-A3B	`local-qwen35-35b`	131072	8	`qwen3_coder`
Qwen3.5-9B	`local-qwen35-9b`	131072	8	`qwen3_coder`
Gemma4-31B	`local-gemma4-31b`	131072	8	`gemma4`
GLM-4.5-Air	`local-glm45-air`	131072	8	`hermes`

• **API access.** GPT-5.5 and GPT-5.4 were accessed through the OpenAI API. Claude Sonnet 4.6 was accessed through the Anthropic API. DeepSeek-V4-Pro was accessed through the DeepSeek API.

• **Local deployment.** The Qwen, Gemma, and GLM models were served locally through vLLM (Kwon et al., 2023) on a single A800 node with 8 NVIDIA A800-SXM4-80GB GPUs, CUDA 12.8, and vLLM 0.19.1. Models were loaded in `torch.bfloat16` with tensor parallel size 8, matching the 8 visible A800 GPUs. GPU memory utilization was set to 0.90 by default, while maximum context length was selected per model as listed in Table 4. vLLM automatic tool choice was enabled, with a model family specific parser used to normalize tool calls before they reached the harness. Claude Code runs used the Anthropic compatible endpoint exposed by vLLM, while Codex, Terminus-2, and DocTools used its OpenAI-compatible endpoint. Qwen3.6 and Gemma4 Codex runs additionally used explicit chat templates. Server-side decoding followed the harness request parameters, while the serving process fixed model loading, context length, parallelism, and tool-call parsing.

G Harness Configuration Details

All harnesses are executed through the same Harbor task lifecycle. For each task, Harbor builds or reuses the task Docker image, mounts the task workspace, launches the selected agent inside an isolated environment, collects the submitted artifact, and then runs the deterministic verifier. This design keeps the task files, agent runtime, and verifier separated: the agent edits only the workspace it receives, while the verifier evaluates the final artifact after the agent exits.

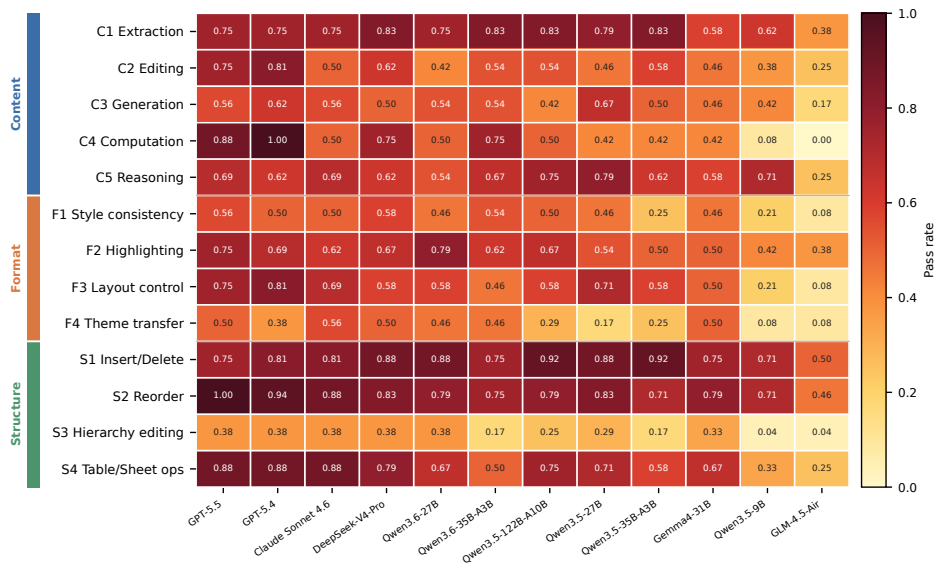


Figure 12: Atomic operation accuracy by model on L1 tasks for all evaluated models. Scores are averaged over the available harness settings for each model.

Docker task images. Every task contains an `environment/Dockerfile`. The Dockerfile starts from a harness-specific base image, copies the source documents into the container, and installs task-local document skills when required. Benchmark scripts preflight the base image before execution, ensuring that a configuration fails early if the expected CLI or image is missing.

• **Codex.** Codex runs use dedicated container images with the corresponding harness versions and required dependencies preinstalled for frontier-model and open-weight configurations. The harness provides an open shell, file-system access, and script execution. Skill and no-skill settings differ only in whether document skills are materialized into the task image and exposed to the agent.

• **Claude Code.** Claude Code runs use dedicated container images with the corresponding harness version and dependencies preinstalled. The harness provides an interactive coding-agent loop with terminal execution and file-system feedback. For vLLM-backed runs, adaptive thinking is disabled so that open-weight models operate through the same external tool loop without a provider-specific reasoning budget.

• **Terminus-2.** Terminus-2 exposes a stateful shell with full file-system feedback. We use a 200-turn limit, a JSON action parser, and temperature 0.2. Terminus-2 does not receive document-specific skills, it measures what a compact shell agent can accomplish through general programming and file inspection alone.

• **DocTools.** DocTools runs use a dedicated container image with the corresponding harness implementation and dependencies preinstalled. The harness exposes structured document-editing tools rather than an unrestricted shell. We use a temperature of 0.0 and a small fixed step budget, allowing us to evaluate a document-specific tool interface separately from the broader flexibility of programmable shell environments.

Skill and no-skill materialization. Skill-enabled tasks copy the same released document skills into the task image and expose them through task metadata. No-skill runs are produced by a deterministic materialization script: it copies the benchmark project, removes skill directory declarations, strips skill advertisements from task instructions, removes Dockerfile skill-copy steps, and deletes task-local skill folders. As a result, w/ skill and w/o skill settings share the same source documents, verifiers, task structure, and harness interface, procedural document skills are the only controlled difference.

Table 5: Skill-induced pass-rate changes by model, difficulty level, and operation family. Values are computed as w/ skill minus w/o skill accuracy; green indicates gains and red indicates regressions. Bold marks the largest gain and largest regression within each harness block and column.

Model	Overall	L1	L2	L3	L4	C	F	S
Claude Code								
Claude Sonnet 4.6	**-0.033**	+0.040	**-0.175**	**-0.017**	-0.017	**-0.036**	**-0.045**	**-0.044**
DeepSeek-V4-Pro	-0.033	**-0.060**	-0.100	+0.050	**-0.050**	-0.012	-0.015	-0.032
Gemma4-31B	+0.067	+0.100	+0.075	**+0.067**	+0.033	**+0.073**	-0.008	**+0.076**
Qwen3.6-27B	+0.038	+0.000	+0.050	+0.033	**+0.067**	+0.055	+0.038	+0.051
Qwen3.6-35B-A3B	+0.010	+0.020	+0.025	+0.017	-0.017	+0.000	+0.000	+0.006
Qwen3.5-122B-A10B	-0.005	-0.020	-0.025	+0.033	-0.017	+0.012	+0.023	-0.025
Qwen3.5-27B	**+0.071**	+0.040	**+0.275**	+0.033	+0.000	+0.061	**+0.083**	+0.051
Qwen3.5-35B-A3B	+0.033	+0.020	+0.050	+0.050	+0.017	+0.036	+0.045	+0.025
Qwen3.5-9B	-0.005	-0.020	+0.000	-0.017	+0.017	-0.006	+0.008	-0.006
GLM-4.5-Air	+0.038	**+0.140**	+0.025	+0.017	-0.017	+0.018	+0.030	+0.006
Codex								
GPT-5.5	**+0.024**	+0.020	+0.075	+0.050	**-0.033**	+0.012	+0.015	**+0.025**
GPT-5.4	+0.024	+0.040	+0.075	+0.000	+0.000	+0.030	+0.030	+0.013
DeepSeek-V4-Pro	+0.024	+0.060	+0.025	**-0.017**	**+0.033**	+0.030	-0.008	+0.019
Gemma4-31B	+0.014	+0.060	+0.000	+0.000	+0.000	-0.012	+0.008	+0.006
Qwen3.6-27B	**-0.038**	**-0.120**	**-0.175**	**+0.083**	+0.000	**-0.030**	+0.008	**-0.013**
Qwen3.6-35B-A3B	+0.014	-0.100	+0.100	+0.033	+0.033	**+0.048**	**-0.015**	+0.013
Qwen3.5-122B-A10B	-0.010	+0.020	-0.075	+0.017	-0.017	-0.006	+0.000	-0.013
Qwen3.5-27B	+0.024	+0.000	+0.100	+0.017	+0.000	+0.012	+0.015	+0.025
Qwen3.5-35B-A3B	+0.019	+0.020	**+0.125**	-0.017	-0.017	+0.006	**+0.038**	+0.000
Qwen3.5-9B	+0.024	**+0.100**	-0.025	+0.017	+0.000	+0.000	+0.015	+0.025

H HARNESS COST ACCOUNTING

Figure 2(b) reports average runtime and token usage at the harness level. For all harnesses, runtime is taken directly from Harbor timing records, which measure each task from start to termination and are independent of harness-specific token accounting. This timing includes agent interaction, tool execution, file operations, verifier invocation, and timeout duration when applicable. We report *s/task* as the mean seconds per task across valid settings for each harness, and *min/run* as the corresponding full-run time.

Token usage counts prompt/input and completion/output tokens across the full interaction trajectory. Unlike runtime, token accounting may differ across harnesses because different runtimes expose different usage metadata. Specifically:

- **DocTools.** We use the agent-reported usage fields when available. For trajectories where the LLM-call records preserve the visible interaction but do not expose reliable token-usage metadata, we estimate token usage from the visible LLM-call trajectory and calibrate it against DocTools runs with reliable usage accounting.

- **Terminus-2.** We compute token usage directly from the recorded usage fields over the full interaction trajectory, without additional calibration.

- **Codex.** Some Codex trajectories preserve the full visible interaction history but do not expose reliable server-side usage fields through the runtime interface. For these cases, we estimate token usage from the visible Codex trajectory and calibrate it against Codex runs with reliable token accounting. Values marked with † in Figure 2(b) denote these calibrated token estimates.

- **Claude Code.** We compute token usage directly from the recorded usage fields over the full interaction trajectory, without additional calibration.

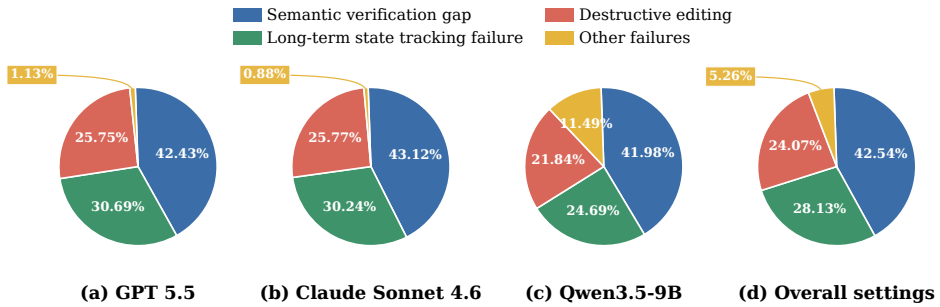


Figure 13: Distribution of failure modes across representative settings and the overall aggregate.

To keep harness averages comparable, each row is computed only over model settings for which the corresponding harness produced valid run records under the reported experimental configuration. Settings without valid results are excluded because including them would mix unavailable or invalid executions with actual harness cost, thereby distorting the runtime and token averages.

I Full operation-level capability map.

Figure 12 reports the complete operation-level heatmap for all evaluated models. Each cell gives the average pass rate on L1 atomic tasks for one model and one atomic operation, averaged over the available harness settings for that model. This full view complements the representative heatmap in the main text and confirms that the same bottleneck operations recur across the broader model set. In particular, operations requiring document-native state control, such as hierarchy editing, theme transfer, computation, and style consistency, remain substantially less reliable than localized content access or object movement operations.

J Full Skill-Injection Results

Table 5 reports the full skill-injection deltas for Claude Code and Codex. We report the overall deltas by difficulty level (L1 to L4) and the deltas by operation family: Content (C), Format (F) and Structure (S). Positive values indicate that enabling skills improves pass rate on the corresponding subset, while negative values indicate regressions. Zero values indicate that the with-skill and no-skill settings achieve the same pass rate on that subset. Within each harness block and column, boldface marks the largest positive and largest negative delta, highlighting the strongest gains and regressions caused by skill injection.

K Quantitative Failure-Mode Analysis

We extract verifier-grounded signals from failed runs and group them by failure mode. Figure 13 reports the distributions across representative settings and the overall aggregate. Semantic verification gaps are the most frequent category, while long-term state tracking failures and destructive editing also account for substantial shares of failures.

Table 6: Full DocOps 210 task list. Input and output types are normalized file-format labels for the source artifacts and required submission.

Task ID	Level	Operation labels	Input types	Output type
excel_001	L1	C4 Computation	Excel	Excel
excel_002	L1	C4 Computation	Excel	Excel
excel_003	L1	F2 Highlighting	Excel	Excel
excel_004	L1	S4 Table/Sheet operations	Excel	Excel
excel_005	L1	C5 Reasoning	Excel	
excel_006	L1	C3 Generation	Excel	Excel
excel_007	L1	C1 Extraction	Excel	
excel_008	L1	C2 Editing	Excel	Excel
excel_009	L1	F1 Style consistency	Excel	Excel
excel_010	L1	F3 Layout control	Excel	Excel
excel_011	L1	F4 Theme transfer	Excel	Excel
excel_012	L1	S1 Insert/Delete	Excel	Excel
excel_013	L1	S2 Reorder	Excel	Excel
excel_014	L1	S3 Hierarchy editing	Excel	Excel
pdf_001	L1	C1 Extraction	PDF	
pdf_002	L1	C5 Reasoning	PDF	
pdf_003	L1	F2 Highlighting	PDF	PDF
pdf_004	L1	S2 Reorder	PDF	PDF
pdf_005	L1	C2 Editing	PDF	PDF
pdf_006	L1	C3 Generation	PDF	PDF
pdf_007	L1	F1 Style consistency	PDF	PDF
pdf_008	L1	F3 Layout control	PDF	PDF
pdf_009	L1	F4 Theme transfer	PDF	PDF
pdf_010	L1	S1 Insert/Delete	PDF	PDF
pdf_011	L1	S3 Hierarchy editing	PDF	PDF
pdf_012	L1	S4 Table/Sheet operations	PDF	PDF
ppt_001	L1	C3 Generation	PPT	PPT
ppt_002	L1	F3 Layout control	PPT	PPT
ppt_003	L1	F4 Theme transfer	PPT	PPT
ppt_004	L1	S2 Reorder	PPT	PPT
ppt_005	L1	S4 Table/Sheet operations	PPT	PPT
ppt_006	L1	S1 Insert/Delete	PPT	PPT
ppt_007	L1	C1 Extraction	PPT	
ppt_008	L1	C2 Editing	PPT	PPT
ppt_009	L1	C5 Reasoning	PPT	
ppt_010	L1	F1 Style consistency	PPT	PPT
ppt_011	L1	F2 Highlighting	PPT	PPT
ppt_012	L1	S3 Hierarchy editing	PPT	PPT
word_001	L1	S3 Hierarchy editing	Word	Word
word_002	L1	S1 Insert/Delete	Word	Word
word_003	L1	C2 Editing	Word	Word
word_004	L1	F1 Style consistency	Word	Word
word_005	L1	C1 Extraction	Word	
word_006	L1	S2 Reorder	Word	Word
word_007	L1	C3 Generation	Word	Word
word_008	L1	C5 Reasoning	Word	
word_009	L1	F2 Highlighting	Word	Word
word_010	L1	F3 Layout control	Word	Word
word_011	L1	F4 Theme transfer	Word	Word
word_012	L1	S4 Table/Sheet operations	Word	Word
excelc_001	L2	C5 Reasoning; C2 Editing; F2 Highlighting	Excel	Excel

Task ID	Level	Operation labels	Input types	Output type
excelc_002	L2	S4 Table/Sheet operations; C4 Computation; F4 Theme transfer	Excel	Excel
excelc_003	L2	S2 Reorder; C3 Generation	Excel	Excel
excelc_004	L2	S3 Hierarchy editing; F3 Layout control	Excel	Excel
excelc_005	L2	C1 Extraction; F2 Highlighting	Excel	Excel
excelc_006	L2	C2 Editing; F1 Style consistency	Excel	Excel
excelc_007	L2	S1 Insert/Delete; C3 Generation	Excel	Excel
excelc_008	L2	S2 Reorder; F4 Theme transfer	Excel	Excel
excelc_009	L2	C1 Extraction; S4 Table/Sheet operations	Excel	Excel
excelc_010	L2	C4 Computation; F2 Highlighting	Excel	Excel
pdfc_001	L2	C5 Reasoning; F2 Highlighting	PDF	PDF
pdfc_002	L2	S2 Reorder; S3 Hierarchy editing	PDF	PDF
pdfc_003	L2	C3 Generation; F4 Theme transfer	PDF	PDF
pdfc_004	L2	S4 Table/Sheet operations; F2 Highlighting	PDF	PDF
pdfc_005	L2	C1 Extraction; C3 Generation	PDF	PDF
pdfc_006	L2	C2 Editing; F1 Style consistency	PDF	PDF
pdfc_007	L2	S1 Insert/Delete; F4 Theme transfer	PDF	PDF
pdfc_008	L2	S2 Reorder; C3 Generation	PDF	PDF
pdfc_009	L2	C5 Reasoning; C2 Editing	PDF	PDF
pdfc_010	L2	F4 Theme transfer; F3 Layout control	PDF	PDF
pptc_001	L2	F4 Theme transfer; F1 Style consistency; F3 Layout control	PPT	PPT
pptc_002	L2	S2 Reorder; C3 Generation	PPT	PPT
pptc_003	L2	C5 Reasoning; F2 Highlighting	PPT	PPT
pptc_004	L2	S4 Table/Sheet operations; C2 Editing	PPT	PPT
pptc_005	L2	C1 Extraction; F2 Highlighting	PPT	PPT
pptc_006	L2	C2 Editing; F1 Style consistency	PPT	PPT
pptc_007	L2	S1 Insert/Delete; C3 Generation	PPT	PPT
pptc_008	L2	S3 Hierarchy editing; F3 Layout control	PPT	PPT
pptc_009	L2	S2 Reorder; F4 Theme transfer	PPT	PPT
pptc_010	L2	C1 Extraction; S4 Table/Sheet operations	PPT	PPT
wordc_001	L2	S3 Hierarchy editing; F1 Style consistency	Word	Word
wordc_002	L2	S2 Reorder; C3 Generation	Word	Word
wordc_003	L2	C5 Reasoning; F2 Highlighting	Word	Word
wordc_004	L2	S4 Table/Sheet operations; C2 Editing	Word	Word
wordc_005	L2	C1 Extraction; F2 Highlighting	Word	Word
wordc_006	L2	C2 Editing; F1 Style consistency	Word	Word
wordc_007	L2	S1 Insert/Delete; C3 Generation	Word	Word
wordc_008	L2	S2 Reorder; F4 Theme transfer	Word	Word
wordc_009	L2	C1 Extraction; S4 Table/Sheet operations	Word	Word
wordc_010	L2	C5 Reasoning; C2 Editing	Word	Word
excelwf_001	L3	S4 Table/Sheet operations; C4 Computation; S2 Reorder; F2 Highlighting; F4 Theme transfer	Excel	Excel
excelwf_002	L3	S2 Reorder; F3 Layout control; C4 Computation; F2 Highlighting; S1 Insert/Delete	Excel	Excel
excelwf_003	L3	C4 Computation; S1 Insert/Delete; S2 Reorder; F2 Highlighting; F4 Theme transfer	Excel	Excel
excelwf_004	L3	S4 Table/Sheet operations; C4 Computation; F1 Style consistency; S2 Reorder	Excel	Excel
excelwf_005	L3	S1 Insert/Delete; S2 Reorder; C4 Computation; F4 Theme transfer; F1 Style consistency	Excel	Excel
pdfwf_001	L3	S2 Reorder; S1 Insert/Delete; C3 Generation; S3 Hierarchy editing	PDF	PDF
pdfwf_002	L3	F4 Theme transfer; C3 Generation; F3 Layout control; S2 Reorder	PDF	PDF
pdfwf_003	L3	S1 Insert/Delete; S2 Reorder; S3 Hierarchy editing; C3 Generation	PDF	PDF
pdfwf_004	L3	S2 Reorder; C3 Generation; S3 Hierarchy editing; F4 Theme transfer	PDF	PDF
pdfwf_005	L3	S2 Reorder; S3 Hierarchy editing; F4 Theme transfer; C3 Generation	PDF	PDF
pptwf_001	L3	S2 Reorder; S1 Insert/Delete; C3 Generation; F1 Style consistency; F4 Theme transfer	PPT	PPT

Task ID	Level	Operation labels	Input types	Output type
pptwf_002	L3	F4 Theme transfer; F1 Style consistency; S3 Hierarchy editing; C2 Editing	PPT	PPT
pptwf_003	L3	S1 Insert/Delete; S4 Table/Sheet operations; F3 Layout control; F4 Theme transfer	PPT	PPT
pptwf_004	L3	C2 Editing; S2 Reorder; F3 Layout control; F1 Style consistency	PPT	PPT
pptwf_005	L3	C2 Editing; S2 Reorder; F1 Style consistency; F3 Layout control	PPT	PPT
wordwf_001	L3	C2 Editing; S2 Reorder; S3 Hierarchy editing; F1 Style consistency; S1 Insert/Delete	Word	Word
wordwf_002	L3	S2 Reorder; S3 Hierarchy editing; F3 Layout control; F1 Style consistency	Word	Word
wordwf_003	L3	C2 Editing; F1 Style consistency; S4 Table/Sheet operations	Word	Word
wordwf_004	L3	S3 Hierarchy editing; S4 Table/Sheet operations; C3 Generation; C5 Reasoning; F1 Style consistency	Word	Word
wordwf_005	L3	C2 Editing; C3 Generation; F1 Style consistency; F3 Layout control; S1 Insert/Delete; S3 Hierarchy editing; S4 Table/Sheet operations	Word	Word
excelxr_001	L4	C1 Extraction; C5 Reasoning; S2 Reorder; S4 Table/Sheet operations	Excel, PDF, Word	Excel
excelxr_002	L4	C1 Extraction; C4 Computation; F2 Highlighting; S4 Table/Sheet operations	Excel, PDF, Word	Excel
excelxr_003	L4	C1 Extraction; C5 Reasoning; S4 Table/Sheet operations	Excel, PDF, Word	Excel
excelxr_004	L4	C1 Extraction; C2 Editing; S4 Table/Sheet operations	Excel, PDF, Word	Excel
excelxr_005	L4	C1 Extraction; S3 Hierarchy editing; S4 Table/Sheet operations	Excel, PDF, Word	Excel
pdfxr_001	L4	C1 Extraction; C3 Generation; S1 Insert/Delete; S2 Reorder; S3 Hierarchy editing	PDF, Word	PDF
pdfxr_002	L4	C1 Extraction; C2 Editing; S1 Insert/Delete; S2 Reorder; S3 Hierarchy editing	Excel, PDF, Word, PNG	PDF
pdfxr_003	L4	C1 Extraction; C5 Reasoning; S1 Insert/Delete; S2 Reorder; S3 Hierarchy editing	Excel, PDF, Word, PNG	PDF
pdfxr_004	L4	C1 Extraction; C2 Editing; S1 Insert/Delete; S2 Reorder; S3 Hierarchy editing	Excel, PDF, Word, PNG	PDF
pdfxr_005	L4	C1 Extraction; C5 Reasoning; S2 Reorder; S3 Hierarchy editing	Excel, PDF, Word, PNG	PDF
pptxr_001	L4	C1 Extraction; C4 Computation; F1 Style consistency; S1 Insert/Delete; S4 Table/Sheet operations	Excel, PPT, Word, PNG	PPT
pptxr_002	L4	C1 Extraction; C2 Editing; F1 Style consistency; S1 Insert/Delete; S2 Reorder; S4 Table/Sheet operations	Excel, PPT, Word, PNG	PPT
pptxr_003	L4	C1 Extraction; C2 Editing; C3 Generation; F1 Style consistency; S1 Insert/Delete; S4 Table/Sheet operations	Excel, PPT, PDF, Word, PNG	PPT
pptxr_004	L4	C1 Extraction; C4 Computation; F1 Style consistency; S1 Insert/Delete; S4 Table/Sheet operations	Excel, PPT, Word, PNG	PPT
pptxr_005	L4	C1 Extraction; C2 Editing; C3 Generation; F1 Style consistency; F4 Theme transfer; S1 Insert/Delete	Excel, PPT, PDF, Word	PPT
wordxr_001	L4	C1 Extraction; C2 Editing; F1 Style consistency; S3 Hierarchy editing; S4 Table/Sheet operations	Excel, PDF, Word, PNG	Word
wordxr_002	L4	C1 Extraction; C2 Editing; C5 Reasoning; F1 Style consistency; S4 Table/Sheet operations	Excel, PDF, Word	Word
wordxr_003	L4	C1 Extraction; C2 Editing; F1 Style consistency; F2 Highlighting; S3 Hierarchy editing; S4 Table/Sheet operations	Excel, PDF, Word, PNG	Word
wordxr_004	L4	C1 Extraction; C2 Editing; F1 Style consistency; S4 Table/Sheet operations	Excel, PDF, Word	Word
wordxr_005	L4	C1 Extraction; C2 Editing; F1 Style consistency; F2 Highlighting; S3 Hierarchy editing; S4 Table/Sheet operations	Excel, PDF, Word, PNG	Word
L3_v2_011	L3	C1 Extraction; C2 Editing; C3 Generation; C4 Computation; C5 Reasoning; F1 Style consistency; F2 Highlighting; F3 Layout control; S1 Insert/Delete; S2 Reorder; S3 Hierarchy editing; S4 Table/Sheet operations	Excel	Excel
L3_v2_012	L3	C2 Editing; S2 Reorder	Excel	Excel
L3_v2_013	L3	C2 Editing; S2 Reorder	Excel	Excel
L3_v2_014	L3	C2 Editing; C4 Computation; S4 Table/Sheet operations	Excel	Excel
L3_v2_015	L3	C2 Editing; C4 Computation; S4 Table/Sheet operations	Excel	Excel
L3_v2_016	L3	C1 Extraction; C2 Editing; C3 Generation; C4 Computation; C5 Reasoning; F1 Style consistency; F2 Highlighting; F3 Layout control; F4 Theme transfer; S1 Insert/Delete; S2 Reorder; S3 Hierarchy editing; S4 Table/Sheet operations	Excel	Excel

Task ID	Level	Operation labels	Input types	Output type
L3_v2_017	L3	C2 Editing; S2 Reorder	Excel	Excel
L3_v2_018	L3	C2 Editing; S2 Reorder	Excel	Excel
L3_v2_019	L3	C2 Editing; C4 Computation; S4 Table/Sheet operations	Excel	Excel
L3_v2_020	L3	C1 Extraction; C2 Editing; C3 Generation; C4 Computation; F1 Style consistency; F2 Highlighting; F3 Layout control; F4 Theme transfer; S1 Insert/Delete; S2 Reorder; S3 Hierarchy editing; S4 Table/Sheet operations	Excel	Excel
L3_v2_031	L3	C2 Editing; C4 Computation; F1 Style consistency; S4 Table/Sheet operations	PDF	PDF
L3_v2_032	L3	C2 Editing; S2 Reorder	PDF	PDF
L3_v2_033	L3	C2 Editing; S2 Reorder	PDF	PDF
L3_v2_034	L3	C2 Editing; C4 Computation; F1 Style consistency; S4 Table/Sheet operations	PDF	PDF
L3_v2_035	L3	C2 Editing; S2 Reorder	PDF	PDF
L3_v2_036	L3	C2 Editing; S2 Reorder	PDF	PDF
L3_v2_037	L3	C2 Editing; C4 Computation; F1 Style consistency; S4 Table/Sheet operations	PDF	PDF
L3_v2_038	L3	C2 Editing; C4 Computation; F1 Style consistency; S4 Table/Sheet operations	PDF	PDF
L3_v2_039	L3	C2 Editing; C4 Computation; F1 Style consistency; S4 Table/Sheet operations	PDF	PDF
L3_v2_040	L3	C2 Editing; C4 Computation; F1 Style consistency; S4 Table/Sheet operations	PDF	PDF
L3_v2_021	L3	C2 Editing; C4 Computation; F1 Style consistency; S4 Table/Sheet operations	PPT	PPT
L3_v2_022	L3	C2 Editing; S2 Reorder	PPT	PPT
L3_v2_023	L3	C2 Editing; S2 Reorder	PPT	PPT
L3_v2_024	L3	C2 Editing; S2 Reorder	PPT	PPT
L3_v2_025	L3	C2 Editing; S2 Reorder	PPT	PPT
L3_v2_026	L3	C2 Editing; C4 Computation; F1 Style consistency; S4 Table/Sheet operations	PPT	PPT
L3_v2_027	L3	C2 Editing; C4 Computation; F1 Style consistency; S4 Table/Sheet operations	PPT	PPT
L3_v2_028	L3	C2 Editing; C4 Computation; F1 Style consistency; S4 Table/Sheet operations	PPT	PPT
L3_v2_029	L3	C1 Extraction; C2 Editing; C3 Generation; C4 Computation; C5 Reasoning; F1 Style consistency; F3 Layout control; S1 Insert/Delete; S2 Reorder; S3 Hierarchy editing	PPT	PPT
L3_v2_030	L3	C1 Extraction; C2 Editing; C3 Generation; C4 Computation; F1 Style consistency; F2 Highlighting; F3 Layout control; F4 Theme transfer; S1 Insert/Delete; S2 Reorder; S3 Hierarchy editing; S4 Table/Sheet operations	PPT	PPT
L3_v2_001	L3	C2 Editing; C4 Computation; F1 Style consistency; F3 Layout control; S3 Hierarchy editing; S4 Table/Sheet operations	Word	Word
L3_v2_002	L3	C2 Editing; C4 Computation; F1 Style consistency; F3 Layout control; S3 Hierarchy editing; S4 Table/Sheet operations	Word	Word
L3_v2_003	L3	C2 Editing; C4 Computation; F1 Style consistency; F3 Layout control; S3 Hierarchy editing; S4 Table/Sheet operations	Word	Word
L3_v2_004	L3	C2 Editing; C4 Computation; F1 Style consistency; F3 Layout control; S3 Hierarchy editing; S4 Table/Sheet operations	Word	Word
L3_v2_005	L3	C2 Editing; S2 Reorder; S3 Hierarchy editing; S1 Insert/Delete	Word	Word
L3_v2_006	L3	C2 Editing; S2 Reorder; S3 Hierarchy editing	Word	Word
L3_v2_007	L3	C2 Editing; S2 Reorder; S3 Hierarchy editing	Word	Word
L3_v2_008	L3	C2 Editing; S2 Reorder; S3 Hierarchy editing	Word	Word
L3_v2_009	L3	C1 Extraction; C2 Editing; C3 Generation; C5 Reasoning; F1 Style consistency; F2 Highlighting; F3 Layout control; S1 Insert/Delete; S2 Reorder; S3 Hierarchy editing	Word	Word
L3_v2_010	L3	C1 Extraction; C2 Editing; C3 Generation; C4 Computation; C5 Reasoning; F1 Style consistency; F2 Highlighting; F3 Layout control; F4 Theme transfer; S1 Insert/Delete; S2 Reorder; S3 Hierarchy editing; S4 Table/Sheet operations	Word	Word

Task ID	Level	Operation labels	Input types	Output type
L4_v2_042	L4	C1 Extraction; C3 Generation; C4 Computation; C5 Reasoning; F2 Highlighting; S1 Insert/Delete; S3 Hierarchy editing; S4 Table/Sheet operations	Excel, PDF, Word	Excel, PDF
L4_v2_043	L4	C1 Extraction; C2 Editing; C3 Generation; C4 Computation; C5 Reasoning; F1 Style consistency; F2 Highlighting; S1 Insert/Delete; S2 Reorder; S4 Table/Sheet operations	Excel, PPT, PDF, Word	Excel, PPT
L4_v2_044	L4	C1 Extraction; C4 Computation; F2 Highlighting; S1 Insert/Delete; S4 Table/Sheet operations	Excel, PDF, Word	Excel
L4_v2_080	L4	C1 Extraction; C2 Editing; C3 Generation; C5 Reasoning; F1 Style consistency; F2 Highlighting; S2 Reorder; S3 Hierarchy editing; S4 Table/Sheet operations	Excel, PDF, Word	Excel, PDF
L4_v2_045	L4	C1 Extraction; C2 Editing; C3 Generation; C5 Reasoning; F1 Style consistency; F2 Highlighting; S1 Insert/Delete; S3 Hierarchy editing	Excel, PDF, Word	PDF, Word
L4_v2_046	L4	C1 Extraction; C2 Editing; C3 Generation; C5 Reasoning; F1 Style consistency; F2 Highlighting; S1 Insert/Delete; S3 Hierarchy editing	Excel, PDF, Word	PDF, Word
L4_v2_049	L4	C1 Extraction; C2 Editing; C3 Generation; C4 Computation; C5 Reasoning; F2 Highlighting; S3 Hierarchy editing; S4 Table/Sheet operations	Excel, PDF, Word	Excel, PDF, Word
L4_v2_050	L4	C1 Extraction; C2 Editing; C3 Generation; C4 Computation; C5 Reasoning; S1 Insert/Delete; S3 Hierarchy editing; S4 Table/Sheet operations	Excel, PDF, Word	Excel, PDF, Word
L4_v2_051	L4	C1 Extraction; C2 Editing; C3 Generation; C4 Computation; C5 Reasoning; F1 Style consistency; F2 Highlighting; S2 Reorder; S4 Table/Sheet operations	Excel, PDF, Word	Excel, PDF
L4_v2_052	L4	C1 Extraction; C3 Generation; C4 Computation; C5 Reasoning; F2 Highlighting; S1 Insert/Delete; S2 Reorder; S3 Hierarchy editing; S4 Table/Sheet operations	Excel, PDF, Word	Excel, PDF
L4_v2_053	L4	C1 Extraction; C4 Computation; C5 Reasoning; S4 Table/Sheet operations	Excel, PDF, Word	Excel, PDF
L4_v2_054	L4	C1 Extraction; C4 Computation; C5 Reasoning; S4 Table/Sheet operations	Excel, PDF, Word	Excel, PDF
L4_v2_055	L4	C1 Extraction; C2 Editing; S1 Insert/Delete; S2 Reorder; S3 Hierarchy editing	Excel, PDF, Word	PDF
L4_v2_056	L4	C1 Extraction; C2 Editing; C3 Generation; C5 Reasoning; F1 Style consistency; S1 Insert/Delete; S2 Reorder; S3 Hierarchy editing; S4 Table/Sheet operations	Excel, PPT, PDF, Word	PPT, Word
L4_v2_057	L4	C1 Extraction; C2 Editing; C3 Generation; C5 Reasoning; F1 Style consistency; S1 Insert/Delete; S2 Reorder; S3 Hierarchy editing	Excel, PPT, PDF, Word	PPT, PDF
L4_v2_058	L4	C1 Extraction; C3 Generation; C4 Computation; C5 Reasoning; F1 Style consistency; S1 Insert/Delete; S2 Reorder; S4 Table/Sheet operations	Excel, PPT, PDF, Word	Excel, PPT
L4_v2_059	L4	C1 Extraction; C2 Editing; C3 Generation; C4 Computation; C5 Reasoning; F1 Style consistency; S1 Insert/Delete; S2 Reorder; S4 Table/Sheet operations	Excel, PPT, PDF, Word	Excel, PPT
L4_v2_060	L4	C1 Extraction; C2 Editing; C3 Generation; C4 Computation; C5 Reasoning; F1 Style consistency; F2 Highlighting; S1 Insert/Delete; S2 Reorder; S4 Table/Sheet operations	Excel, PPT, PDF, Word	Excel, PPT
L4_v2_061	L4	C1 Extraction; C4 Computation; C5 Reasoning; S4 Table/Sheet operations	Excel, PDF, Word	Excel, PPT
L4_v2_062	L4	C1 Extraction; C4 Computation; C5 Reasoning; S4 Table/Sheet operations	Excel, PPT, PDF, Word	Excel, PPT
L4_v2_041	L4	C1 Extraction; C4 Computation; C5 Reasoning; S4 Table/Sheet operations	Excel, PDF, Word	Excel, PDF, Word
L4_v2_047	L4	C1 Extraction; C4 Computation; C5 Reasoning; S4 Table/Sheet operations	Excel, PDF, Word	Excel, Word
L4_v2_048	L4	C1 Extraction; C4 Computation; C5 Reasoning; S4 Table/Sheet operations	Excel, PDF, Word	Excel, Word
L4_v2_063	L4	C1 Extraction; C2 Editing; C3 Generation; C5 Reasoning; F1 Style consistency; F2 Highlighting; S1 Insert/Delete; S3 Hierarchy editing	Excel, PDF, Word	PDF, Word
L4_v2_064	L4	C1 Extraction; C2 Editing; C3 Generation; C4 Computation; C5 Reasoning; F2 Highlighting; S1 Insert/Delete; S3 Hierarchy editing; S4 Table/Sheet operations	Excel, PDF, Word	PDF, Word
L4_v2_065	L4	C1 Extraction; C2 Editing; C3 Generation; C5 Reasoning; F1 Style consistency; F2 Highlighting; S2 Reorder; S3 Hierarchy editing	Excel, PPT, PDF, Word	PPT, Word
L4_v2_066	L4	C1 Extraction; C2 Editing; C3 Generation; C4 Computation; C5 Reasoning; F1 Style consistency; F2 Highlighting; S3 Hierarchy editing; S4 Table/Sheet operations	Excel, PDF, Word	Excel, Word

Task ID	Level	Operation labels	Input types	Output type
L4_v2_067	L4	C1 Extraction; C2 Editing; C3 Generation; C4 Computation; C5 Reasoning; F2 Highlighting; S1 Insert/Delete; S3 Hierarchy editing; S4 Table/Sheet operations	Excel, PDF, Word	Excel, Word
L4_v2_068	L4	C1 Extraction; C2 Editing; C3 Generation; C4 Computation; C5 Reasoning; F2 Highlighting; S3 Hierarchy editing; S4 Table/Sheet operations	Excel, PDF, Word	Excel, Word
L4_v2_069	L4	C1 Extraction; C2 Editing; C3 Generation; C4 Computation; C5 Reasoning; F2 Highlighting; S3 Hierarchy editing; S4 Table/Sheet operations	Excel, PDF, Word	Excel, Word
L4_v2_070	L4	C1 Extraction; C2 Editing; C3 Generation; C4 Computation; C5 Reasoning; F2 Highlighting; S3 Hierarchy editing; S4 Table/Sheet operations	Excel, PDF, Word	Excel, Word
L4_v2_071	L4	C1 Extraction; C2 Editing; C3 Generation; C4 Computation; C5 Reasoning; F2 Highlighting; S3 Hierarchy editing; S4 Table/Sheet operations	Excel, PDF, Word	Excel, Word
L4_v2_072	L4	C1 Extraction; C2 Editing; C3 Generation; C4 Computation; C5 Reasoning; F2 Highlighting; S3 Hierarchy editing; S4 Table/Sheet operations	Excel, PDF, Word	Excel, Word
L4_v2_073	L4	C1 Extraction; C2 Editing; C3 Generation; C4 Computation; C5 Reasoning; F2 Highlighting; S3 Hierarchy editing; S4 Table/Sheet operations	Excel, PDF, Word	Excel, Word
L4_v2_074	L4	C1 Extraction; C4 Computation; C5 Reasoning; S4 Table/Sheet operations	Excel, PDF, Word	Excel, Word
L4_v2_075	L4	C1 Extraction; C4 Computation; C5 Reasoning; S4 Table/Sheet operations	Excel, PDF, Word	Excel, Word
L4_v2_076	L4	C1 Extraction; C4 Computation; C5 Reasoning; S4 Table/Sheet operations	Excel, PDF, Word	Excel, Word
L4_v2_077	L4	C1 Extraction; C4 Computation; C5 Reasoning; S4 Table/Sheet operations	Excel, PDF, Word	Excel, Word
L4_v2_078	L4	C1 Extraction; C4 Computation; C5 Reasoning; S4 Table/Sheet operations	Excel, PDF, Word	Excel, Word
L4_v2_079	L4	C1 Extraction; C2 Editing; C3 Generation; C5 Reasoning; F1 Style consistency; F2 Highlighting; S1 Insert/Delete; S2 Reorder; S3 Hierarchy editing; S4 Table/Sheet operations	Excel, PDF, Word	Excel, Word